



# **BASICISTA KONEKIELEEN**

## **Konekieliopas 6502-prosessorille**

Commodore 64  
VIC-20  
Apple II, IIe, IIfx  
Atari 600 XL, 800 XL, 130 XE

Eskoensio Pipatti

BITTI **BITTI**-kirjat  
Tecnopress Oy



**Basicista konekieleen  
Konekieliopas 6502-prosessorille**

Eskoensio Pipatti

Kim Leidenius, Ataria käsittelevät osuudet  
Aki Korhonen, Applea käsittelevät osuudet

Kirjassa olevat ohjelmat on tarkastettu huolella. Emme kuitenkaan voi taata niiden virheettömyyttä emmekä vastaa esiintyneistä virheistä.

© Copyright Tecnopress Oy  
Kaikki oikeudet pidätetään.

Tämän teoksen tekstiosuuden osittainenkin kopioiminen on tekijänoikeuslain (404/61, muut. 897/80) ja valokuvain (405/61, muut. 898/80) mukaisesti kielletty ilman kustantajan kirjallista suostumusta lukuun ottamatta tätä teosta koskevia arvosteluja ja katsauksia.

Kirjapaino Lars Eriksen Oy Espoo 1986  
ISBN 951-832-006-3

## **MikroASSEMBLER-OHJELMA ON SAATAVANA MYÖS LEVYKKEELLE TALLENNETTUNA**

Tätä tilauskorttia käyttämällä voit tilata kirjassa listatun MikroASSEMBLER-ohjelman valmiiksi levykkeelle tallennettuna.

**Hinta on 79 mk + lähetyskulut 18 mk**

Voit tehdä tilauksen kätevästi repäisemällä tämän sivun irti seläkekierteestä, täyttämällä kääntöpuolen tilauskortin ja pistämällä sen postiin. MikroBITTI maksaa postimaksun puolestasi.

---

**Tilaa Commodore 64:n peliohjelmoinnin perusteos!**

### **HUVIA JA HYÖTYÄ, COMMODORE 64**

Huvia ja hyötyä -kirja on pelejä ohjelmoivan Commodoristin tietoteos. Se selvittää pelihahmojen teon itse määritellyillä merkeillä, spriteillä ja monivärispriteillä. Kirja kertoo myös, miten pelihahmot saa liikkumaan, miten tehdään taustat ja miten taustakuvaa scrollataan eri suuntiin. Huvia ja hyötyä neuvoa myös äänitehosteiden synnyttämisessä ja oman musiikin säveltämisessä.

**Lisäksi kirjassa on listattuna 10 peliohjelmaa ja 6 hyötyohjelmaa Commodore 64:lle!**

**Teoksen hinta on 115 mk + lähetyskulut 18 mk**

Kätevimmin saat kirjan omaksesi repäisemällä tämän kaavakkeen irti seläkekierteestä, täyttämällä kääntöpuolen tilauskortin ja pistämällä sen postiin. MikroBITTI maksaa postimaksun puolestasi.



☐ Tilaan BASICISTA KONEKIELEEN-  
kirjan sisältämän MikroASSEMBLER-  
ohjelman valmiiksi levykkeelle tallennettuna  
hintaan 79 mk + lähetyskulut 18 mk.  
(vain CBM-64, VIC-20, Apple, Atari)

Mikroni merkki on:

☐ Commodore 64    ☐ VIC-20  
☐ Apple            ☐ Atari

nimi \_\_\_\_\_

osoite \_\_\_\_\_

postitoimipaikka \_\_\_\_\_

Vastauslähetys  
Hki 38 Lupa 394

**Lukijapalvelu**  
**PL 64**  
**00003 HELSINKI**

☐ Tilaan HUVIA JA HYÖTYÄ  
COMMODORE 64 -kirjan hintaan 115 mk +  
lähetyskulut 18 mk.

nimi \_\_\_\_\_

osoite \_\_\_\_\_

postitoimipaikka \_\_\_\_\_

Vastauslähetys  
Hki 38 Lupa 394

**Lukijapalvelu**  
**PL 64**  
**00003 HELSINKI**

# Sisällysluettelo

|                                                          |           |
|----------------------------------------------------------|-----------|
| <b>KONEKIELI KIEHTOO</b>                                 | <b>9</b>  |
| <b>1. KIVINEN TIE: Basicista konekieleen</b>             | <b>11</b> |
| <b>2. MURSKATAANPA BITTEJÄ: Lukujärjestelmät</b>         | <b>15</b> |
| Heksadesimaaliluvut                                      | 15        |
| Kaksijärjestelmä                                         | 16        |
| 8 bittiä on tavu                                         | 18        |
| <b>3. TAVAT JA TOTTUMUKSET: Työskentely konekielellä</b> | <b>23</b> |
| Kolme kirjainta ei aina ole IBM                          | 26        |
| Heksalataaja                                             | 27        |
| Basiclataaja                                             | 28        |
| Assemblerit                                              | 29        |
| <b>4. KOHTI YDINTÄ: 6502:n rakenne</b>                   | <b>31</b> |
| Tunne tietokoneesi                                       | 31        |
| Pääportista sisään                                       | 33        |
| Akku on kiireinen                                        | 35        |
| Juoksupojat X ja Y                                       | 37        |
| Ohjelmalaskuri ja käskyrekisteri                         | 39        |
| Osoitteet pinoon                                         | 41        |
| Liput salkoon                                            | 43        |
| <b>5. POSTI KULKEE: Osoitusmuodot</b>                    | <b>49</b> |
| Välitön (Immediate)                                      | 50        |
| Absoluuttinen (Absolute)                                 | 51        |
| Nollasivu (Zero Page)                                    | 51        |
| Absoluuttinen,X ja absoluuttinen,Y                       | 52        |
| Nollasivu,X (Zero Page,X)                                | 54        |
| Nollasivu,Y (Zero Page,Y)                                | 54        |
| Epäsuora,Y (Indirect,Y)                                  | 55        |
| Epäsuora,X (Indirect,X)                                  | 56        |
| Epäsuora (Indirect)                                      | 58        |
| Suhteellinen (Relative)                                  | 59        |
| Sisältyvä (Implied) ja akku (Accumulator)                | 62        |

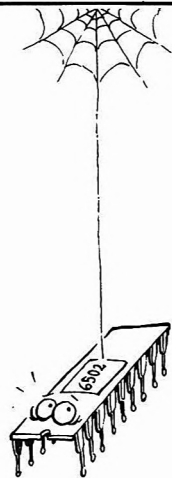
|                                                                                                                                |            |
|--------------------------------------------------------------------------------------------------------------------------------|------------|
| <b>6. ITSE ASIAAN: 6502:n käskykanta</b> .....                                                                                 | <b>65</b>  |
| Siirtokäskyt LDA, STA, LDX, STX, LDY, STY,<br>PHA, PLA, PHP, PLP, TAX, TXA,<br>TAY, TYA, TSX, TXS .....                        | 66         |
| Laskenta ja logiikka ADC, SBC, AND, EOR, ORA, BIT,<br>CMP, CPX, CPY, ASL, LSR, ROL,<br>ROR, DEC, DEX, DEY, INC, INX, INY ..... | 71         |
| Haarautumiset ja hyyt JMP, JSR, BNE, BEQ, BCC, BCS,<br>BMI, BPL, BVC, BVS, BRK, RTI, RTS .....                                 | 83         |
| Lippukäskyt CLC, CLD, CLI, CLV, SEC, SED, SEI .....                                                                            | 90         |
| Tyhjäntoimittaja NOP .....                                                                                                     | 92         |
| <br><b>7. ISOMPI PALA: Konekielinen ohjelmointi</b> .....                                                                      | <b>93</b>  |
| Pistelaskuri .....                                                                                                             | 95         |
| Tarkastaja .....                                                                                                               | 99         |
| Assemblerin käyttö .....                                                                                                       | 102        |
| <br><b>8. COMMODORE 64</b> .....                                                                                               | <b>105</b> |
| Muistikartta .....                                                                                                             | 105        |
| Vapaat alueet .....                                                                                                            | 107        |
| Haetaan näppäimiltä .....                                                                                                      | 108        |
| Luetaan peliohjainta .....                                                                                                     | 110        |
| Piirrämme kuvaruudulle .....                                                                                                   | 111        |
| Tarkkuusgrafiikka .....                                                                                                        | 115        |
| Muistiosoitteet .....                                                                                                          | 117        |
| Hyödyllisiä ROM-rutiineja .....                                                                                                | 118        |
| <br><b>9. VIC-20</b> .....                                                                                                     | <b>121</b> |
| Muistikartta .....                                                                                                             | 121        |
| Vapaat alueet .....                                                                                                            | 123        |
| Luetaan peliohjainta .....                                                                                                     | 124        |
| Piirrämme kuvaruudulle .....                                                                                                   | 125        |
| Haetaan näppäimistöltä .....                                                                                                   | 127        |
| Muistiosoitteet .....                                                                                                          | 129        |
| Hyödyllisiä ROM-rutiineja .....                                                                                                | 131        |
| <br><b>10. APPLE</b> .....                                                                                                     | <b>133</b> |
| Muistikartta .....                                                                                                             | 133        |
| Vapaat alueet .....                                                                                                            | 135        |
| Haetaan näppäimiltä .....                                                                                                      | 135        |
| Luetaan peliohjainta .....                                                                                                     | 136        |

|                                  |                                                                      |
|----------------------------------|----------------------------------------------------------------------|
| <b>Piirrämme kuvaruudulle</b>    | 136                                                                  |
| <b>Tarkkuusgrafiikka</b>         | 140                                                                  |
| <b>Muistiosoitteet</b>           | 142                                                                  |
| <b>Hyödyllisiä ROM-rutiineja</b> | 143                                                                  |
| <b>11. ATARI</b>                 | <b>145</b>                                                           |
| Muistikartta                     | 145                                                                  |
| Vapaat alueet                    | 147                                                                  |
| Haetaan näppäimiltä              | 148                                                                  |
| Luetaan peliohjainta             | 150                                                                  |
| Piirrämme kuvaruudulle           | 151                                                                  |
| Grafiikan hallinta ja Antic      | 153                                                                  |
| Anticin käskykanta               | 155                                                                  |
| Syöttö ja tulostus               | 157                                                                  |
| Muistiosoitteet                  | 161                                                                  |
| <b>LIITTEET</b>                  |                                                                      |
| <b>LIITE I:</b>                  | 6502:n käskyhakemisto 165                                            |
| <b>LIITE II:</b>                 | Lukujärjestelmämuunnokset ja<br>käskyjen pikahakemisto 189           |
| <b>LIITE III:</b>                | MikroASSEMBLER 197                                                   |
|                                  | Commodore 64 206                                                     |
|                                  | VIC-20 214                                                           |
|                                  | APPLE 231                                                            |
|                                  | ATARI 238                                                            |
| <b>LIITE IV:</b>                 | Ohjelmien syöttöohjeet ja<br>Tarkastajan käyttö (C-64 ja VIC-20) 245 |
| <b>LIITE V:</b>                  | Hakemisto 249                                                        |





# Konekieli kiehtoo



Konekieli ohjelmointi kiehtoo kaikkia kotitietokoneen omistajia. Nykyisissä kotitietokoneissa on jo lähes kaikissa sisäänrakennettu Basic-kieli ja koneen mukana seuraa hyvät oppaat ja perusteokset tämän maailman yleisimmän ohjelmointikielen omaksumiseksi. Basic on erittäin hyvä ohjelmointikieli laitteeseen tutustuttaessa ja ensimmäisiä ohjelmia rakenneltaessa. Se sisältää kaiken mitä tarvitaan, jotta kone saadaan tekemään halutut asiat.

Basicilla on myös huonot puolensa. Se muodostaa eräänlaisen muurin käyttäjän ja koneen välille. Koneen sisäiset toiminnot ja mahdollisuudet jäävät Basiciin tyytyvälle täysin tuntemattomiksi. Lisäksi Basic on joissakin tehtävissä liian hidas. Animaatioiden ja hyvien kuvaruutupelien rakentelu pelkällä Basicilla ei onnistu riittävän hyvin.

Konekielen kiehtovuus piilee juuri siinä, että sen avulla koneen salat avautuvat käyttäjälle. Viimeisetkin sopukat koneen rakenteessa tulevat sen myötä tutuiksi ja ohjelmiin tulee runsaasti uutta puhtia.

Tämä kirja on perusteos, joka johdattaa Basicia jo opetelleen harrastajan konekielen monimutkaiseen, mutta ehdottomasti mielenkiintoiseen maailmaan. Tämän kirjan antaman tiedon omaksuminen ei edellytä, että hallitsisit Basic-ohjelmointia edes hyvin. Riittää, kun tiedät, mitä on ohjelmointi ja miten Basicin puolella konetta "käsketään".

1. luvun tarkoituksena on pala palalta romuttaa Basiciin tottuneiden aivojen mukavuuksilla höystetty ajatusmaailma ja avata uusi näkökulma tietokoneeseen. Tämä uusi sähköinen ajatusmaailma sisältää mikropiirejä, rekistereitä ja lukusarjoja.

2. luku perehdyttää uusiin lukujen merkitsemistapoihin, jotka on syytä ottaa

---

takataskuun mukaan, kun lähdemme käsky kerrallaan avaamaan konekielen portteja.

3. luku kertoo konekieliohjelmoinnin apuvälineistä, ohjelmista ja menettelytavoista, jotka tekevät konekielen yhtä helpoksi ohjelmoida kuin mikä tahansa muukin kieli.

4.luku käsittelee 6502 (ja 6510) -prosessorin rakennetta. Konekielikäskyt manipuloivat aina jotain prosessorin sisäisistä rekistereistä (muistipaikoista), joten ensimmäistäkään konekielen käskyä ei voi ymmärtää ilman tämän kappaleen tietoutta.

5. ja 6. kappaleesta alkaa vasta se asia, jonka takia olet tämän kirjan käteesi ottanut — konekielikäskyjen läpikäynti. Malttamattomimmat ovat jo selanneet kirjaa niiltä sivuilta, mutta muista: ilman neljän ensimmäisen luvun pohjatietoa konekielen opiskelu on kuin veteen piirtäisi.

Kirjan lopussa on listattuna MikroASSEMBLER-ohjelma, jota tulet käyttämään edetessäsi esimerkistä toiseen kirjaa lukiessasi. Jos koneesi on jokin muu kuin C-64, käännä esimerkkejä tehdessäsi kirja auki myös omaa konettasi käsittelevän luvun kohdalta. Saat sieltä tiedot, miten esimerkit tehdään Sinun koneellasi.

Jälkikäteen koet varmasti erittäin tärkeiksi kirjan liitteet, joissa konekielikäskyt on listattu sekä aakkosjärjestyksessä että heksa- ja desimaalilukujen mukaisessa järjestyksessä. Samaten lukujärjestelmien väliset muunnokset käyvät liitteitten avulla helposti.

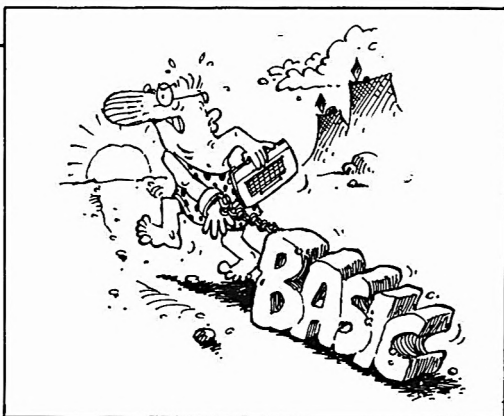
Tämän kirjan tietous riittää varsin pitkälle konekieliharrastuksessa. Jos haluat vielä syvällisempää tietoa, vilkaise merkkikohtaista kirjallisuusluetteloa. Sieltä löydät oman konemerkkisi salaisuuksia paljastavaa luettavaa, valitettavan usein tosin englanninkielistä.

Sysmässä 6. marraskuuta 1985

*Eskoensio Pipatti*

Basicista konekieleen

# 1



## Kivinen tie

Konekieli koostuu peräkkäisistä käskyistä aivan kuten niin sanotut korkeamman tason kieletkin, joihin Basic lukeutuu. Konekielen ohjelmointiympäristö, eli käytettävissä olevien käskyjen suorittamat toiminnot, koneen antama tuki ohjelmointityölle jne poikkeaa niin paljon esimerkiksi Basicista, ettei pelkkien käskyjen opetteleminen riitä. Ensin on käytävä läpi muutama ohjelmointiympäristöön liittyvä perusasia.

### Muuttujat muistissa

Basicilla ohjelmoiva on tottunut siihen, että hänen ei tarvitse pitää huolta siitä, missä päin muistia mikäkin ohjelman käsky tai muuttujaan varastoitu tieto kulloinkin sijaitsee. Kaiken tämän hoitaa Basic-tulkki omien taulukoidensa avulla. Basicissa voidaan varastoida lukuja kirjaimien taakse:

10 LET A = 5

\*

\*

120 PRINT A

Konekielessä se ei ole mahdollista. Kaikki varastoitava tieto on vietävä johonkin tiettyyn muistipaikkaan ja tarvittaessa haettava samasta paikasta. Periaatteessa jokainen muistipaikka vastaa konekielessä muuttujaa.

Basicisti on tottunut yhdellä käskyllä hoitamaan kuvaruututulostusta, laskutoimituksia suurillakin luvuilla, sisäänsyöttöä näppäimistöltä, tallennusta kasetille tai levyille.

Konekielellä se ei ole näin yksinkertaista. Konekielen käskyt tekevät vain vähän kerrallaan. Jokainen toiminto on koottava pienistä palasista ja jokainen basicin

käskey onkin konekielen puolella jo pieni ohjelman pätkä.

### Tavuittain

Konekieli käsittelee tavuja, 8-bittisiä kaksijärjestelmän lukuja. Yhteen tavuun voi tallentaa korkeintaan luvun 255. Se tuntuu pieneltä Basicin liukulukuihin tottuneesta ohjelmoijasta. Mutta sillä pärjää. Yllättävän harvoin käytännön ohjelmis-  
sa tarvitaan suurempia lukuja.

| bitti 7 | bitti 6 | bitti 5 | bitti 4 | bitti 3 | bitti 2 | bitti 1 | bitti 0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| 1       | ∅       | 1       | 1       | ∅       | ∅       | ∅       | 1       |

Konekielisen ohjelman käskyt ovat tavun mittaisia, vaikkakin niillä voi olla yhden tai kahden tavun pituisia lisämääreitä (esimerkiksi haettavan luvun muistosoite).

### Käyttöjärjestelmä tukena

Ohjelmoija ei silti ole aivan yksin konekielen käskyjen kanssa. Käyttöjärjestelmä ja itse Basic-tulkkin antavat suosiolla osan omistaan: konekielisissä ohjelmissa voi käyttää tehokkaasti hyväkseen niitä valmiita pieniä konekielirutiineja, jotka jokaisessa koneessa löytyvät ROM-muistista.

Kuvaruututulostus, näppäimistön seuraaminen ja etenkin kasetti- ja levyope-  
raatiot sujuvat varsin kivuttomasti näiden valmiiden palikoiden avulla. Kirjan  
lopussa olevissa konekohtaisissa luvuissa näitä rutiineja käsitellään tarkemmin.

Itse asiassa kaikki Basicin käskyt kuten monimutkaiset laskutoimitukset, oh-  
jelmien tallennukset ja lataamiset, merkkien hakeminen näppäimistöltä, merk-  
kien tulostaminen kuvaruudulle, jne — kaikki ne on tavoitettavissa myös kone-  
kielestä käsin.

Kun tämän kirjan ensimmäisten lukujen myötä opit pala palalta konekielen  
yksinkertaiset käskyt, saatat pitää toivottomana kasata niistä monimutkaisia oh-  
jelmia. Niin perin juurin vähäisiä tuntuvat olevan ne toiminnot, joita yksi käsky  
tekee.

Älä kuitenkaan tuskastu. Konekielinen ohjelmointi on kuitenkin käytännössä  
etupäässä koneen ROM-muistista löytyvien valmisrutiinien jatkuvaa hyväksikäyt-  
töä. Niitä kutsutaan konekielessä samanlaisella käskyllä kuin Basicissa kutsutaan  
aliohjelmiä. Konekielen kutsukäsky on nimeltään JSR (Jump to Subroutine =  
hyppää aliohjelmaan) ja se vastaa täysin Basicin käskyä GOSUB.

## Ohjelmien sisäänsyöttö

Basic-ohjelmien syöttäminen tietokoneeseen on erittäin yksinkertaista. Kun kytket koneeseesi virran päälle, ilmestyy ruutuun alkutekstiä, joka mm. kertoo käytettävissä olevan muistin määrän, ja kursori alkaa vilkkua seuraavalla rivillä kertoen koneen olevan valmis ottamaan vastaan käskysi. Annat koneelle rivinumerolla varustetun ohjelmarivin ja kone tallentaa sen muistiin ja on valmis ottamaan vastaan seuraavan rivin.

Konekieltä et kuitenkaan voi syöttää muistiin samalla tavalla. Luvussa 3 kerrotaan, miten voit piilottaa konekieliohjelmiasi Basicin DATA-lauseisiin tai miten saat yksinkertaisella Basic-ohjelmalla syötettyä muistiin konekielen käskyjä. Yleensä konekieliohjelmoinnissa tarvitaan kuitenkin monimutkaisempi ohjelma, jonka avulla saat ohjelmiasi tallennettua koneen muistiin. Tällainen ohjelma, MikroASSEMBLER, on listattuna kirjan liitteessä kolme.

## Ohjelmien sijoitus

Basic-ohjelmaa koneeseen syöttäessäsi sinun ei tarvitse miettiä, mihin kohtaan vapaata muistialuetta ohjelmiasi sijoittuvat. Basic-tulkilla on tietty muistiosoite, josta alkaen se sijoittaa antamasi käskyrivit sellaisenaan merkki merkiltä (käskysanat tokeneiksi lyhennettynä). Ohjelmaa ajettaessa Basic-tulkki löytää ohjelman aina tästä samasta kohdasta.

Konekielellä ohjelmoidessasi sinun on kuitenkin itse päätettävä, mistä muistiosoitteesta haluat ohjelmiasi alkavan. Se edellyttää koneen muistikartan hallitsemista, jotta tiedät mistä löytyy riittävästi vapaata tilaa. Oman koneesi muistikartan löydät luvuista 8-11.

Ohjelmatilan lisäksi sinun on myös varattava muistialue niille muuttujille, joita ohjelmassasi käytät. Tässä muuttujalla tarkoitetaan pikemminkin lukujen väliarastointipaikkaa. Kuhunkin muistipaikkaan voit varastoida yhden luvun. Yleensä näille muistipaikoille varataan tilaa juuri konekieliohjelman edeltä tai niiden jälkeen.

## Konekieliohjelman käynnistäminen

Basic-ohjelman käynnistät yksinkertaisesti komennolla RUN. Konekieliohjelmien käynnistäminen on yhtä yksinkertaista. Jos ohjelmiasi on muistissa ja Basicin kursori vilkkuu ruudulla, voit käynnistää sen komennolla SYS XXXX (C-64 ja VIC-20), CALL XXXX (Apple) tai USR (XXXX) (Atari).

Jos olet syöttänyt konekielisen ohjelman koneeseen assembler-ohjelmalla ja olet edelleen assemblerissa, voit käynnistää ohjelmiasi yleensä komennolla .G (joka ei siis ole Basic-komento, vaan assemblerin ymmärtämä ohjauskomento). Eri koneiden levykäyttöjärjestelmät tarjoavat muitakin mahdollisuuksia käynnistykseen.





Murskataanpa bittejä

2



# Kymmenen sormea

Arkielämässä käytämme laskiessamme kymmenjärjestelmää, jonka kantaluku on kymmenen. Se tarkoittaa sitä, että kun yhdeksikön jälkeen lisäämme lukuun vielä yhden, emme voi kuvata lukua enää yhdellä numerolla vaan meidän on merkattava viereiseen "sarakkeeseen" ykkönen sen merkiksi, että luvussa on kymmeniä yksi kappaletta.

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11,...

Me käytämme kymmenjärjestelmää sen vuoksi, että meillä sattuu olemaan yhteensä kymmenen sormea. Sormet muodostivat esi-isiemme ensimmäisen version helmitaulusta.

Kun kymmenjärjestelmässä näitä kymmeniä tulee niitäkin 10 kappaletta, merkkäämme edelleen uuteen sarakkeeseen ykkösen sen merkiksi, että satoja on yksi kappale. Historia ei kerro, käyttivätkö esi-isämme varpaitaan kymmenluku-  
jen muistamiseen.

## Heksadesimaaliluvut

Kymmenjärjestelmä on jokaisella peruskoulun läpikäyneellä niin vireen syöpy-  
nyt, että on vaikea kuvitella mitään muuta menetelmää.

Valitettavasti tietokoneiden rakenne on suurelta osin teknisistä syistä jouduttu tekemään sellaiseksi, että lukujen välittäminen ja käsitteleminen kymmenjärjes-  
telmässä on sen sisällä hankalaa. Sen vuoksi on otettu käyttöön muita lukujen

merkitsemistapoja, joista heksadesimaalinen järjestelmä esiintyy vahvasti konekielisen ohjelmoinnin yhteydessä.

Heksajärjestelmän kantaluku on kuusitoista. Se merkitsee, että yhdellä numerolla on pystyttävä esittämään kaikki (10-järjestelmän) luvut viiteentoista asti. Vasta kuudennentoista kohdalla merkitsemme viereiseen sarakkeeseen ykkösen. Eli se numero, joka kymmenjärjestelmässä ilmaisi, kuinka monta 'kymmentä' luvussa on, ilmaisee heksajärjestelmässä, kuinka monta 'kuuttatoista' luvussa on.

Ylimääräisten numeroiden ongelma on aikanaan ratkaistu kätevästi siten, että numerot, joita kymmenjärjestelmässä kuvattaisiin luvuilla 10-15, merkitään kirjaimilla A, B, C, D, E, ja F. Vasta F:n jälkeen tulee heksajärjestelmässä luku 10H. Se tarkoittaa kymmenjärjestelmässä lukumäärää 16.

**0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F, <sup>10</sup>10, 11,...**

Heksalukujen perään merkitään iso H-kirjain, jottei niitä sekoitettaisi kymmenjärjestelmän lukuihin. Konekielen yhteydessä myös luvun edessä oleva \$-merkki osoittaa luvun olevan heksajärjestelmässä.

Heksajärjestelmä saattaa aluksi tuntua vaikealta hahmottaa mielekkäästi, mutta kuvittelepa, jos ihmisillä olisi 8 sormea kummassakin kädessä. Kummalla järjestelmällä luulisit meidän koulussa käsittelevän lukuja?

Heksajärjestelmässä lukujen kasvaminen tapahtuu vastaavalla logiikalla kuin kymmenjärjestelmässä. Aina kuudentoista tultua täyteen ykkösten sarakkeessa (F+1), lisätään toisen sarakkeen 'kuusiatoista' yhdellä. Ykkösten sarakkeessa aloitetaan uudestaan nollasta. Kun nämä kuudettoista lisääntyvät lukumäärältään kuuteentoista asti, merkitään kolmanteen sarakkeeseen ykkönen (100H). Kymmenjärjestelmässä se merkitsi sataa, mutta heksaluvuissa se merkitsee 16 kertaa 16 eli 256 näin kymmenjärjestelmän kannalta katsottuna.

## Kaksijärjestelmä

Kaksijärjestelmä on varsin hauska. Siinä kantaluku on kaksi, joten jo ykkösen jälkeen saavuttaessa lukuun kaksi on meidän merkittävä viereiseen sarakkeeseen ykkönen. Kakkoseen asti ei siis koskaan päästä.

**0, 1, 10, 11,...**

Kaikki luvut koostuvat vain ykkösistä ja nollista. Käsiemme käyttökelpoisuuden kannalta on melkein sama, ovat ne sitten rakentuneet 10- tai 16-järjestelmän mukaan, mutta miten olisi, jos pohjana olisi kaksijärjestelmä?

Kaksijärjestelmä eli binaarilukujärjestelmä on tietokoneen varsinainen oma järjestelmä. Tietokoneet on rakennettu siten, että niiden virtapiirit voivat olla vain kahdessa tilassa.

Useimmissa elektronisissa piireissä tämä toteutetaan siten, että johto on joko jännitteellinen tai jännitteetön. Jännitteetöntä tilaa merkitään nolalla (0) ja jännitteellistä tilaa ykkösellä (1). Yksi johto virtapiiriin sisällä voi siis kuvata ja välittää yhden kaksijärjestelmän numeron.

Jos asetamme kahdeksan johtoa rinnakkain, voimme niiden avulla kuvata ja välittää muille piireille kahdeksannumeroisen kaksijärjestelmän luvun, esimerkiksi 10011001. Elektroniset piirit voidaan tehdä sellaisiksi, että ne osaavat laskea näiden kahdeksan johdon välittämällä binaariluvuilla ja välittää tulokset eteenpäin toisilla johdoilla.

Binaariluvut voidaan muuttaa kymmenjärjestelmän luvuiksi helposti. Jokainen ykkönen vastaa tiettyä kahden potenssia. Vähiten merkitsevä bitti vastaa nollatta potenssia eli ykköstä, toinen bitti kakkosen ensimmäistä potenssia eli kakkosta, kolmas bitti kakkosen toista potenssia eli nelosta jne. Nämä luvut vain lasketaan yhteen ja nollat unohdetaan.

Kahdeksanbittisessä luvussa bittien arvot ovat seuraavat:

|               | bit7  | bit6  | bit5  | bit4  | bit3  | bit2  | bit1  | bit0  |
|---------------|-------|-------|-------|-------|-------|-------|-------|-------|
|               | 1     | 1     | 1     | 1     | 1     | 1     | 1     | 1     |
| 10-järj. luku | 128   | 64    | 32    | 16    | 8     | 4     | 2     | 1     |
| 2:n potenssi  | $2^7$ | $2^6$ | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

Esimerkkilukumme 10011001 olisi siis:

$$128 + 0 + 0 + 16 + 8 + 0 + 0 + 1 = 153$$

Luku 185 taas voidaan muuttaa kaksijärjestelmään seuraavasti: vähennetään siitä 128 ja merkataan bitti7:n kohdalle ykkönen. Jäljelle jäi 57. Siitä emme voi vähentää 64:ää, joten merkkasimme bitti6:n kohdalle nollan. 32 siitä voidaan vähentää, joten merkkasimme bitti5:n kohdalle taas ykkösen. Jäljelle jää tällä kertaa 25. Vähennetään 16 (bitti4 on 1), jolloin jää 9. Vähennämme 8 (bitti3 on 1) ja jää 1. Neljää ja kahta vastaavat bitit ovat nollia, koska niitä ei voida jäännöksestä vähentää ja lopulta bitti1:n kohdalle tulee viimeinen ykkönen. 185 on siis 10111001!

Kaksijärjestelmän lukuja ei juuri koskaan käytetä kuvaamaan tietokoneen lukuja, vaikka se olisi kaikkein selväpiirteisin menetelmä. Kaksijärjestelmän luvut ovat varsin pitkiä, esimerkiksi luku 10 000 on kaksijärjestelmässä

**10011100010000.**

Kaksijärjestelmän luvut eivät myöskään ole havainnollisia.

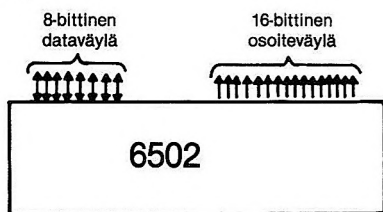
Kaksijärjestelmän sijasta käytetään usein edellä esitettyjä heksalukuja. Joskus käytetään kymmenjärjestelmän lukuja ja varsin harvoin oktaalilukuja (kantalu-kuna 8).

### 8 bittiä on tavu

Nykyisissä kotitietokoneissa käytetään melkein poikkeuksetta 8-bittisiä mikroprosessoreita, jollaisia ovat myös 6502 ja 6510. Mikroprosessori on tietokoneen koko toimintaa ja laskentaa ohjaava elektroninen piiri.

8-bittisyys tarkoittaa, että tietokoneen sisällä luvut kulkevat kahdeksassa rinnakkaisessa johtimessa, joista kukin vastaa yhtä kaksijärjestelmän lukua eli bittiä. Tietokone voi siis yhdellä kertaa siirtää paikasta toiseen 8-numeroisen kaksijärjestelmän luvun.

Osoittaessaan jotain muistipaikkaa, johon tämä 8-bittinen luku on vietävä, mikroprosessori joutuu kuitenkin käyttämään kaksinkertaista pituutta eli 16 bittiä, jotta se pystyisi hallitsemaan riittävän suurta muistitilaa (65536 muistipaikkaa).



*Keskusyksikkö eli mikroprosessori ohjaa ympäristöään kahden väylän avulla. Osoiteväylässä on 16 johdinta ja niillä keskusyksikkö valitsee haluamansa muistipaikan 65536:n joukosta. Dataväylä koostuu 8 johtimesta ja niitä pitkin kulkevat käsiteltävät tiedot ja konekielikäskyt.*

Tätä 65536 muistipaikkaa kutsutaan mikroprosessorin osoitevaruudeksi. Tavallisesti sitä merkitään myös luvulla 64 kilotavua. Miksi 64? Miksei 65? Molemmat merkintätavat ovat oikein, sillä vaikka kilo yleensä tarkoittaa tuhatta yksikköä, niin kilotavu ei ole tuhat tavua vaan 1024 tavua. Se on perinteinen tietokonetekniikan termi ja sen tarkoituksena on yksinkertaistaa puhetta ja tekstiä. Lisäksi 1024 on MELKEIN kuin 1000, eikö vain?

Ammattitietokoneissa on viime aikoina siirrytty käyttämään pääasiassa 16-bittisiä prosessoreita. Niiden elektronisissa piireissä kulkee siis 16 johtoa rinnakkain ja ne voivat käsitellä kerralla enemmän tietoa kuin 8-bittiset prosessorit. Sen vuoksi ne yleensä ovat nopeampia. Ne osoittavat muistiaan yli 20 bitin levyisillä binaariluvuilla, ja niiden osoiteavaruus on usein megatavujen eli miljoonien tavujen luokkaa.

Mutta nyt takaisin omiin 8-bittisiimme. Ne koskevat meitä ja tulevat vielä pitkään olemaan suosituin tyyppi kotitietokoneissa.

8 rinnakkaista bittiä siis välittää kaiken tiedon tietokoneen sisällä. Konekielen käskyt ovat 8-bittisiä eli yhden tavun levyisiä ja ne tiedon palaset, joita näiden käskyjen avulla siirrellään ja muokataan ovat 8 bittiä leveitä. Esimerkiksi käsky, joka siirtää luvun jostain muistipaikasta mikroprosessorin sisälle tiettyyn rekisteriin (muistipaikkaan mikroprosessorin sisällä) on binaarimuodossa

10100101

**Konekieltä** käyttäessäsi et koskaan joudu muistamaan ja syöttämään käskyjä bittimuodossa koneeseen. Ne ovat pitkiä ja vaikeasti muistettavia ja sen vuoksi syötettäessä konekielistä ohjelmaa tietokoneen muistiin käytetään apuohjelmia. Niille voit antaa käskyt mielekkäämmässä muodossa. Tällainen apuohjelma muuttaa antamasi koodit binaariluvuiksi ja sijoittaa ne oikeisiin muistipaikkoihin.

Eräs apuohjelma, jota voi käyttää, on itse Basic-tulkki. Sillähän voi POKEilla lukuja muistiin antamalla ne kymmenjärjestelmässä. Oletko koskaan tullut ajatelleeksi, että POKE-käsky ei sijoita lukujasi muistiin sellaisenaan, kymmenjärjestelmän lukuina. Se muuttaa ne ensin kaksijärjestelmän ykkösiksi ja nolliksi ja vasta sitten suorittaa siirron muistiin.

### Käskyt selväkielisiksi

Usein käskyt ilmaistaan konekielen yhteydessä heksalukuina. Äskeinen käsky olisi heksana \$A5. Käytämme läpi tämän kirjan heksalukujen merkitsemiseen \$-merkkiä, joten aina kun näet luvun edessä \$:n tiedät sen merkitsevän heksajärjestelmän lukua.

Desimaalisena sama käsky olisi 165. Eivätpä tunnu nämäkään luvut sen havainnollisemmilta kuin alkuperäinen käsky, mutta lyhyempiä ne ainakin ovat. Älä kuitenkaan työläänny. Konekielen käskyille on olemassa vielä havainnollisempi tapa, joka esitellään myöhemmin. Ne merkitään kolmen kirjaimen tunnuksella, joka on lyhenne käskyn toimintaa kuvaavasta englanninkielisestä sanasta.

Kun kuitenkin olet utelias, voimme selvittää, mikä on esimerkkikäskyn kirjainlyhenne: se on LDA ja se tarkoittaa 'hae akkuun luku' (englanniksi Load Accumulator, josta lyhenne on peräisin). Akku on mikroprosessorin eniten käytetty rekisteri. Sen kautta suoritetaan yleensä kaikki tiedonsiirrot ja laskenta.

Kun 8-bittinen tavu esitetään heksaluvuilla saavutetaan tietty etu: kaikki 8 bittillä merkittävissä olevat luvut (0 — 255) voidaan esittää kahden heksaluvun avulla. Kymmenjärjestelmällä tarvitsemme suurimmissa luvuissa jo kolme numeroa (100 — 255). Sen vuoksi heksaluvuilla täytetyt taulukot ovat siistejä: sarakkeet ovat aina yhtä leveitä ja vain kahden numeron levyisiä.

Heksaluvuilla on myös toinen hyvä ominaisuus. Jos jaamme 8 bittiä leveän binaariluvun kahtia kahdeksi 4-bittiseksi jonoksi ja muutamme kummankin erikseen heksaluvuksi, on tulos sama kuin jos olisimme muuttaneet suoraan 8-bittisen binaariluvun heksajärjestelmään. Se johtuu siitä, että neljä bittiä voi sisältää luvut 0 — 15 (0000 — 1111) eli juuri samat kuin heksajärjestelmän yksi numero.

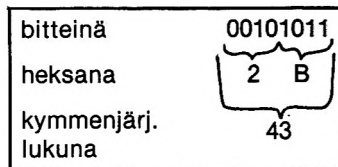
Lisättäessä binaarilukuun 1111 vielä yksi lisää, joudumme ottamaan käyttöön viidennenkin sarakkeen ja tulos on 10000. Heksajärjestelmässä käy samoin, eli luvun \$0F (kymmenjärjestelmässä 15) jälkeen lisättäessä yksi päädyimme lukuun

## Lukujärjestelmät

\$10. Huomaat, että tässä nolla vastaa binaariluvun neljää viimeistä bittiä (kaikki nollia) ja ykkönen binaariluvun viidettä bittiä (ykkönen).

Ohessa on taulukko, jossa on vierekkäin sekä kymmenjärjestelmän, heksajärjestelmän että kaksijärjestelmän vastaavat luvut nollostä eteenpäin. Sen tutkiminen auttaa hahmottamaan, miten luvut käyttäytyvät eri lukujärjestelmissä ja miten ne vastaavat toisiaan. Lisäksi kuvassa on eräs esimerkki tavun ilmaisemisesta biteillä ja miten se voidaan jakaa kätevästi kahteen heksalukuun.

| 8 bitin tavu | Heksaluku | 10-järj. |
|--------------|-----------|----------|
| 00000000     | 00        | 0        |
| 00000001     | 01        | 1        |
| 00000010     | 02        | 2        |
| 00000011     | 03        | 3        |
| 00000100     | 04        | 4        |
| 00000101     | 05        | 5        |
| 00000110     | 06        | 6        |
| 00000111     | 07        | 7        |
| 00001000     | 08        | 8        |
| 00001001     | 09        | 9        |
| 00001010     | 0A        | 10       |
| 00001011     | 0B        | 11       |
| 00001100     | 0C        | 12       |
| 00001101     | 0D        | 13       |
| 00001110     | 0E        | 14       |
| 00001111     | 0F        | 15       |
| 00010000     | 10        | 16       |
| 00010001     | 11        | 17       |
| 00010010     | 12        | 18       |
| ...          | ...       | ...      |
| 00101011     | 2B        | 43       |
| ...          | ...       | ...      |
| ...          | ...       | ...      |
| 11111111     | FF        | 255      |



*Kun kahdeksanbittinen tavu ilmaistaan heksoina, se voidaan jakaa kahteen neljän bitin palaseen ja muuttaa kumpikin erikseen heksaluvuiksi, lopputulos on silti oikea. Tavu voidaan esittää myös kymmenjärjestelmän lukuna 0—255.*

*Muunnostaulukko kaksijärjestelmän, heksadesimaalijärjestelmän ja kymmenjärjestelmän välillä. Heksajärjestelmässä riittää itse asiassa 16 ensimmäistä, kun tavut muunnetaan neljän bitin ryhminä.*

Kirjan lopussa on liitteenä täydellinen taulukko eri lukujärjestelmistä välillä 0 — 255. Nämä ovat juuri oleelliset luvut, koska yhdellä tavulla ei voi kuvata suurempia lukuja. Muistiosoitteita laskiessamme joudumme käsittelemään suurempiakin lukuja eri järjestelmissä. Myös niiden muunnokset lukujärjestelmästä toiseen sujuvat taulukon avulla vaivattomasti.

8-bittisyyden tuoma rajoitettu lukualue 0 — 255 muodostaakin suurimman hankaluuden konekielellä ohjelmoitaessa. Basicin ylellisiin matemaattisiin ominaisuuksiin (liukulukuaritmetiikka, suuri kokonaislukualue) tottuneelle se voi aluksi tuntua kovaakin päänsärkyä aiheuttavalta. Käytännössä se ei kuitenkaan ole niin suuri rajoitus kuin aluksi tuntuu. Hämmästyttävää kyllä, konekielisissä ohjelmissa ei useinkaan tarvita suurempia lukuja. Kaikki konekielen käskyt



(56 kaikkiaan) mahtuvat lisäksi hyvin yhden tavun puitteisiin. Useimmilla käskyillä on monta eri muotoa, jotka liittyvät tapoihin, joilla käskyn vaikutukseen joutuva muistipaikka osoitetaan, joten "vapaita" käskykoodeja ei silti jää montakaan. Peruskäskyjä 6502-prosessorin konekielessä on kuitenkin 56, mikä on suunnilleen saman verran kuin joidenkin koneiden Basic-kielessä.

### **Yhteistyötä Basicin kanssa**

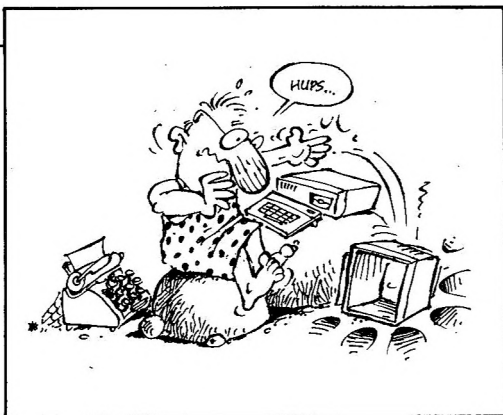
Jos ohjelmointitehtäväsi sisältää suuria lukuja, on sinun rakennettava pieniä aliohjelmia, jotka käsittelevät näitä lukuja tavun mittaisina palasina. Jos taas tehtäväsi sisältää monimutkaista matematiikkaa suurten lukujen lisäksi — esimerkiksi potensseja, sinifunktioita, logaritmeja tai desimaalilukuja — on nämä osat viisainta edelleen ohjelmoida Basicilla ja hoitaa konekielellä kaikki nopeutta vaativat tehtävät. Basic-ohjelmiin voi kaikissa tietokoneissa liittää myös konekielisiä osia. Joissakin tietokoneissa kuten Commodore 64:ssä ja VIC-20:ssä ne käynnistetään Basic-ohjelman sisällä SYS-käskyn avulla, muissa koneissa käytetään CALL-käskyä (Apple) tai USR-funktiota (Atari).



Työskentely konekielellä

# 3

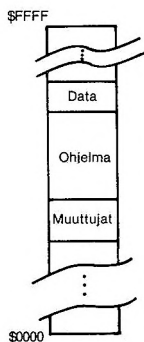
## Tavat ja tottumukset



Olet nyt saanut tuntuman siihen, missä muodossa konekielinen ohjelma sijaitsee tietokoneen muistissa. Se on pitkä jono 8-bittisiä binaarilukuja, jotka eivät ihmiselle sano sellaisenaan mitään. Jonossa olevien lukujen arvon lisäksi ratkaisevaa on myös se, missä ne sijaitsevat.

Kun teet Basic-ohjelmaa, ei sinun tarvitse huolehtia, minne päin muistia sen sijoittaisit. Tämän työn hoitaa automaattisesti koneesi Basic-tulkki.

Konekielellä ohjelmoitaessa on ohjelmoijan päätettävä, mihin muistipaikkoihin ohjelmansa sijoittaa. Se on eräs niistä monista tehtävistä, jotka joutuu konekieleen siirtyessään ottamaan koneelta itselleen. Näihin pulmiin löytyy kuitenkin erilaisia apuneuvoja, joten ajan mittaan ne eivät muodostu rasitteeksi.



*Palanen tietokoneen muistiavaruutta. Konekieliohjelmat sijoitetaan vapaille alueille, joss ne eivät mene päällekkäin koneen muiden toimintojen, esimerkiksi käyttöjärjestelmän, varaamien alueiden kanssa. Konekieliohjelmissa tarvitaan Basic-ohjelmien lailla muuttujia tallennettavia lukuja varten. Konekieliohjelmissa niille on kuitenkin varattava tietty muistialue, jonka osoitteita käytetään tallennettaessa "muuttujaniminä". Muuttuja-alue varataan yleensä joko ohjelmaa välittömästi edeltävistä osoitteista tai ohjelman jälkeen. Vakiotiedoille, esimerkiksi kuvaruudulle tulostettavat tekstit, on myös varattava tilaa. Vakiotiedot kulkevat ohjelman mukana ikäänkuin osana ohjelmakoodia.*

Konekielinen ohjelma on sijoitettava jollekin muistialueelle ja tilaa on varattava myös käsiteltäville luvuille, samalla tavalla kuin Basicilla ohjelmoitaessa käytetään joskus DIM-lausetta. Konekielessä ohjelma ja datat (muuttujat, vakio-luvut ym.) sijaitsevat yhdenvertaisina muistissa. Konekielisen ohjelman käynnistysosoite (SYS-käsky) ja ohjelman kulku pitävät ohjelman pyörimässä omalla alueellaan ilman, että koneen keskusyksikkö harhautuisi data-alueelle ja yrittäisi tulkita niitä käskyiksi.

### Bittikoodit

Jos pääsisimme tunkeutumaan muistipiirien kuoren läpi ja omaisimme "binaariset silmät", näkisimme pitkän jonon 8 bittiä leveitä rekistereitä, eli muistipaikkoja. Jokaisessa olisi taltioituna luku, ei heksana eikä kymmenjärjestelmän luku, vaan binaarisena — ykkösinä ja nollina. Voisimme jollain kohtaa nähdä vaikkapa seuraavanlaisen bittikuvion:

```
*  
*  
*  
10101001  
01000001  
10001101  
00000000  
00000100  
01100000
```

```
*  
*  
*
```

Tämä on osa konekielistä ohjelmaa ja se siirtää mikroprosessorin sisällä olevaan tiettyyn rekisteriin A-kirjainta vastaavan luvun \$41 (muistat, että \$ tarkoitti heksalukua) ja kirjoittaa sen kuvaruudun ensimmäiseen merkkipaikkaan vasempaan yläreunaan (muistiosoitteeseen \$0400, Commodore 64).

Binaariluvut eivät ole selvälukuisia. Ei kukaan muista 8-bittisistä binaariluvuista niiden merkitystä. Mahdollisuuksia on yhteensä 256. Tämän takia konekielistä ohjelmointia ei suoriteta binaariluvuilla, vaikka se on tietokoneen oma esitystapa.

### Heksat

Kuten edellisessä kappaleessa opimme, voimme esittää esimerkin sisältämät binaarikoodit myös heksalukuina, jotka ovat lyhyempiä:

| heksana | 10-järjestelmässä |
|---------|-------------------|
| *       |                   |
| *       |                   |
| *       |                   |
| A9      | 169               |
| 41      | 65                |
| 8D      | 141               |
| 00      | 0                 |
| 04      | 4                 |
| 60      | 96                |
| *       |                   |
| *       |                   |
| *       |                   |

Tästä esitystavasta olisi jo helpompi esimerkiksi kirjan lopussa olevan konekielen käskytaulukon avulla selvittää kunkin tavun merkitys. Ei heksamuoto kuitenkaan selvälukuista tekstiä ole sekään. Se voidaan taulukon avulla kylläkin ymmärtää, mutta suoraan luettavissa ei konekielinen ohjelma ole tälläkään tavalla.

Heksalukujen vieressä ovat samat koodit meille tuttuina 10-järjestelmän lukuina. Basicin POKE-käsky, jolla lukuja sijoitetaan suoraan muistiin, haluaa luvut 10-järjestelmässä. Voit POKEta luvut osoitteisiin 850-855 ja ajaa ohjelman käskyllä SYS 850. Toimi hienosti, eikö vain! Koodi 96 palautti koneen Basiciin pienen pätkämme jälkeen (96 = \$60 = RTS, ReTurn from Subroutine eli palaa aliohjelmasta).

Applen omistajat voivat käynnistää ohjelman komennolla CALL 850. VICissä tarvitsee muuttaa vain luku 4 luvuksi 30.

## Ei kaikki kultaa, mikä kiiltää

Jos yritit katsoa kirjan lopussa olevasta taulukosta heksakoodien merkitystä, menit ehkä hieman ymmällesi. A9 selvisi helposti, se lataa akkurekisteriin jonkin luvun (LDA). Mutta miksi 41? 8D tuntui taas selvältä: STA eli luku akusta muistiin. Mutta sitten taas: 00 eli ohjelman keskeytys!?

Selitys löytyy siitä, etteivät kaikki ohjelman koodit ole käskyjä. Onhan Basicinakin muita kuin käskysanoja. Esimerkiksi PRINT "A" sisältää käskysanan (PRINT) ja sen jälkeen tiedon, mitä tulostetaan ("A").

Samoin konekielessäkin käskykoodeja voi seurata, ja yleensä seuraa, yksi tai kaksi tavua lisäämäreitä eli argumentteja. Käskyä A9 seuraa se luku (\$41), joka akkurekisteriin sijoitetaan. Käskyä 8D seuraa se osoite (\$0400), johon akun sisältämä luku sijoitetaan. Osoite vie kaksi tavua tilaa ja mikroprosessori haluaa, että ne ovat käännettyssä järjestyksessä (00 04).

Ei siis ole yhdentekevää, mistä kohtaa alat konekielistä ohjelmaa tulkita: jos

alat argumentista ja yrität tulkita sen käskyksi, et saa mitään järkevää selville.

Kun tietokoneet olivat uusia laitteita, suoritettiin ohjelmointi tosiaan pelkillä binaariluvuilla! Käskyt syötettiin koneeseen vipukytkimillä, asettamalla kytkimet binaariluvun mukaisesti eri asentoihin ja painamalla nappia, jolloin käsky siirtyi koneen muistiin.

Se oli hankalaa ja aikaa vievää, mutta sen aikaisissa tietokoneissa keskusmuistia oli jopa vähemmän kuin nykyisissä kotitietokoneissa. Pieni muistitila ei sallinut pitää siinä avustavia ohjelmia, jotka olisivat hyväksyneet ohjelmien syötön selväkielisemmässä muodossa. Eikä niissä ollut puhuttakaan välittömästi käytettävissä olevasta ohjelmointikielestä kuten Basicista, jolla nykyajan tietokoneilijat herkkutelevat.

## Kolme kirjainta ei aina ole IBM

On kuitenkin olemassa keino, jolla konekielen käskyt voidaan kirjoittaa ja lukea ilman minkäänlaisten taulukoiden apua. Jokaiselle konekielen käskylle on sovitettu kolmesta kirjaimesta muodostuva tunnus, nimi, joka on lyhenne käskyn toimintaa kuvaavasta englannin kielisestä sanasta tai sanoista. Esimerkiksi LDA (Load Accumulator = lataa akkuun) ja JSR (Jump SubRoutine = hyppää aliohjelmaan).

Tällaiset kolmikirjaimiset käskyt on helppo muistaa, varsinkin kun tuntee niiden takana olevat englannin kieliset sanat.

Edellä ollut A-kirjaimen kuvaruudulle tulostava ohjelman osa olisi näillä kirjainlyhenteillä ilmaistuna seuraavanlainen:

```
LDA #$41
STA $0400
```

Kun tiedämme, että LDA-käsky sijoittaa luvun mikroprosessoriin ja STA-käsky vie sen annettuun muistiosoitteeseen, alkaa ohjelma tuntua jo luettavalta. Älä välitä, vaikka et tässä vaiheessa vielä ymmärtäisikään käskyjen ja merkintöjen tarkoitusta. Opit ne perusteellisesti kirjan seuraavissa luvuissa.

Tietenkään et voi suoraan antaa tietokoneellesi konekielisiä käskyjä näillä kirjainlyhenteillä. Se ei ymmärtäisi niistä tuon taivaallista. Jotta voisit käyttää kirjainlyhenteitä ohjelmoidessasi konekielellä, sinulla on oltava koneessasi pyörimässä ohjelma, joka tulkitsee antamasi kirjainlyhenteet. Se hakee omasta taulukostaan niitä vastaavat binaarikoodit ja sijoittaa vasta ne koneen muistiin haluamaasi kohtaan. Vasta sitten koneesi keskusyksikkö pystyy suorittamaan ohjelmaasi.

Tällaista apuohjelmaa kutsutaan assembleriksi (assembler = kokoaja). Kirjainlyhenteistä muodostuu aivan oma ohjelmointikielensä ja sitä kutsutaan assembly-kieleksi. Assembly-kieli on oikeastaan täysin sama kuin konekieli, mutta merkintätapa on vain sovittu erilaiseksi. Assembler-ohjelmat ovat kuin kielen-

**kääntäjiä:** sanat pysyvät merkitykseltään samoina, mutta assembler kääntää ne ihmisen ymmärtämästä muodosta koneen ymmärtämään muotoon.

## Asento — ojennus!

Heksaluvut ovat varsin käytetty tapa listata ohjelmia esimerkiksi lehden sivuilla, koska ne vievät hyvin vähän tilaa. Tällaista ohjelman listausta kutsutaan ”heksadumpiksi”.

Seuraavassa heksadumpissa on edellä kuvattu esimerkki neljä kertaa peräkkäin, se juntaisi siis A-kirjaimen 4 kertaa kuvaruudun ensimmäiseen merkki-paikkaan. Kuvaruudulle ei tule neljää A-kirjainta.

Basicissa tulostettaessa Basic-tulkki pitää huolta kursorista eli kohdistimesta. Kun tulostamme peräkkäin jotain kuvaruudulle, sijoittaa Basic-tulkki ne myös peräkkäin eli se siirtää kohdistinta sitä mukaa, kun tekstiä tulostetaan. Konekielessä tätä toimintoa ei suoranaisesti ole käytettävissä. Kirjan loppuosassa kerrotaan, miten konekielessä voidaan käyttää hyväksi kursoriohjausta.

Esimerkin voisi korjata lisäämällä osoitteeseen \$0400 aina yhden uudessa tulostuksessa. POKEile ohjelma muistiin ensin listatussa muodossa alkaen samasta osoitteesta kuin äskeisessä esimerkissä. Muuta sitten toinen ohjelmassa esiintyvä numeropari 00 01:ksi, kolmas 02:ksi ja neljäs 03:ksi. Muista, että POKE-käskyyn on laitettava luvut 10-järjestelmässä. Löydät ne edellä olleesta esimerkistä. Viimeinen heksaluku \$60 on 10-järjestelmässä 96 ja se palauttaa koneen kontrollin Basic-tulkille suorituksen jälkeen.

|      |    |    |    |    |    |    |    |    |
|------|----|----|----|----|----|----|----|----|
| 0800 | A9 | 41 | 8D | 00 | 04 | A9 | 41 | 8D |
| 0808 | 00 | 04 | A9 | 41 | 8D | 00 | 04 | A9 |
| 0810 | 41 | 8D | 00 | 04 | 60 |    |    |    |

Heksalistaus koostuu riveistä, joilla ensimmäisenä lukuna on osoite (heksana yleensä sekin) ja sen jälkeen 6-10 heksanumeroparia, joista kukin kuvaa yhden tavun sisältöä. Alussa oleva osoite on rivin ensimmäisen heksanumeroparin osoite muistissa, loput rivillä olevat heksanumeroparit sijoitetaan sitä seuraaviin muistipaikkoihin. Seuraavalla rivillä jatketaan taas siitä, mihin edellisellä rivillä jäätiin.

Vähänkin edellistä esimerkkiä pidemmät ohjelmat käyvät tuskallisiksi syöttää koneeseen POKEilemalla. Silloin tarvitaan jokin apuohjelma, jolle tarvitsee vain antaa lukuarvot ja joka sitten automaattisesti POKEaa ne halutuille paikoille. Tällaisia apuohjelmia on kahdenlaisia: hexloader ja Basicloader eli heksalataaja ja Basiclataaja. Ne eroavat toimintatavoiltaan paljonkin toisistaan.

## Heksalataaja

Heksalataajalla syötetään suoraan koneen muistiin edellisen esimerkkilistauksen mukaista konekieliohjelmia. Heksalistauksessa luvut ovat aina kahden numeron



mittaisia, joten riveistä tulee suoria ja helppolukuisia.

Heksalataaja kysyy ensin alkuosoitteen eli ensimmäiseksi syötettävän tavun osoitteen muistissa. Sen jälkeen lataajalle syötetään heksalukupareja samat 6-10 paria kerrallaan ja siirrytään seuraavan rivin syöttöön.

Jotkut heksalataajaohjelmat tarkistavat automaattisesti sisäänsyötettyjen lukuparien oikeellisuuden. Ne laskevat riveittäin syötetyt luvut yhteen ja rivin lopussa tulostavat tämän summan. Summaa verrataan kopioitavan heksalistauksen rivin lopussa olevaan summaan ja jos luvut olivat samat oli rivi oikein syötetty.

Tällaisen tarkistussummamenettelyn käyttämiseksi on tietenkin jo heksalistauksessa oltava laskettuna lukuparien summat riveittäin valmiiksi ja tulostettuna lukuparien perään. Joka tapauksessa, jos tällaista tarkistussummaa käytetään, ilmenevät virheet jo heti ohjelman syöttövaiheessa ja virheellinen rivi voidaan kirjoittaa uudestaan.

Kun konekielinen ohjelma on syötetty koneen muistiin joko POKEilemalla tai heksalataajaa apuna käyttäen, on ohjelma syytä aivan ensimmäiseksi tallettaa kasetille tai levykkeelle. Sama sääntö koskee myös Basic-ohjelmia. Jos ohjelmassa on virhe, saattaa kone seota ohjelmaa ajettaessa ja silloin vaivalla sisäänsyötetty ohjelma on melkein aina auttamattomasti kadonnut, ainoa keino selvittää tilanteesta kun on virran katkaiseminen.

### Basic-lataaja

Basic-lataaja on varsin mukava tapa säilyttää lyhyitä konekielisiä ohjelmia. Ne voivat olla esimerkiksi suuremman Basic-ohjelman osia, jotka nopeuden takia on kirjoitettu konekielellä.

Basic-lataajassa konekielen käskyt on sijoitettu DATA-lauseisiin ja ohjelmassa on FOR...NEXT -silmukka, joka lukee datat ja POKEaa ne muistiin haluttuihin paikkoihin. Itse asiassa näitä DATA-lauseisiin "piilotettuja" konekieliohjelmia on osana useissa lehtien Basic-ohjelmalistauksissa. Niitä on vaikea erottaa grafiikkamerkkien määrittelyistä, koska ne ovat saman kaltaisia. Ainoa ero on ohjelman jossain kohdassa esiintyvä SYS-käsky, joka käynnistää konekielisen osuuden.

```
10 FOR I=850 TO 855: READ A: POKE I,A: NEXT I  
20 SYS 850: END  
30 DATA 169, 1, 141, 0, 4, 96
```

Basic-lataajaa on helppo käsitellä. Se on puhdas Basic-ohjelma, joten se voidaan ladata ja tallentaa yhdellä kertaa tavallisilla käskyillä eikä erillisiä koodin tallennuksia ja latauksia tarvita. Konekieltä on myös helppo korjata listaamalla Basic-ohjelma ja muuttamalla DATA-lauseiden sisältöä.

Konekieliohjelmien sijoittamisessa DATA-lauseisiin on vain yksi paha puoli:

se vie valtavasti muistitilaa. Yksi konekielen käsky vie DATA-lauseessa keskimäärin 4 merkkiä eli muistitavua: käskyjä esittävät luvut voivat olla joko yksinumeroisia (0-9), kaksinumeroisia (10-99) tai kolminumeroisia (100-255). Niiden lisäksi tulevat vielä lukuja erottavat pilkut, rivinumerot ja käskysana DATA. Suoraan konekielisessä muodossa tallennettaessa yksi käsky tai sen argumenttitavu vie ainoastaan yhden tavun verran tilaa. Pitkät konekieliset ohjelmat eivät siis sovellu DATA-lauseisiin sijoitettaviksi. Pitkien ohjelmien POKEilu muistiin Basic-ohjelman alussa vie lisäksi paljon aikaa.

## Assemblerit

Edellä kuvatut heksalistat ja Basicin DATA-lauseissa piilevät konekieliset ohjelmat ovat vain tapoja konekielisten ohjelmien siirtämiseen paperille kirjoitettuna tai painettuna ja niiden tallentamiseen.

Varsinainen työkalu konekielisessä ohjelmoinnissa on kuitenkin assembler-ohjelma. Sen avulla voit nopeasti ja vaivatta suunnitella ja tehdä itse konekielisiä ohjelmia.

Assemblerilla voit kirjoittaa käskyt ja osoitteet selväkielisessä muodossa ja se huolehtii oikeiden koodien etsimisestä ja sijoittamisesta muistiin. Assemblerille annetaan vain ohjelman alkuosoite ja sen jälkeen ohjelmakäskyjä, lukuja ja osoitteita siinä järjestyksessä kuin ohjelmaa rakennamme. Yhdelle riville kirjoitetaan vain yksi käsky.

Assembler-ohjelmia on hyvin monenlaisia. Yksinkertaisimmat niistä koodittavat käskysi suoraan muistiin sitä mukaa kuin niitä koneelle syötät. Ne pystyvät laskemaan vain hyppyosoitteet taaksepäin (eteenpäin hypättäessähän ei vielä tiedetä, missä osoitteessa hypyn kohde tulee sijaitsemaan).

Hienoimmat assemblerit (assembler-kääntäjät) sallivat antaa käskyille nimiä eli leimoja, jotka kirjoitetaan rivin alkuun ennen varsinaista käskyä. Hypyt eteenkin päin sujuvat antamalla kohdekäskylle nimi, joka sitten kirjoitetaan kyseisen rivin alkuun, kun vastaavaa käskyä ohjelmoidaan. Esimerkiksi seuraavalla tavalla:

|              |                  |                                                                         |
|--------------|------------------|-------------------------------------------------------------------------|
|              | <b>JMP hyppy</b> | (JMP = Jump eli hyppykäsky. Hypätään käskyyn, jonka nimi on 'hyppy')    |
|              | *                |                                                                         |
|              | *                |                                                                         |
|              | *                |                                                                         |
| <b>hyppy</b> | <b>LDA #\$41</b> | (käskyn "nimi" on rivin alussa oleva 'hyppy'. Jatketaan ohjelmaa tästä) |

Tällaisessa tapauksessa assembler-ohjelma käy suunnittelemasi konekielisen ohjelman läpi kaksi kertaa ennen kuin se sijoittaa käskyjä vastaavat koodit muistiin. Ensimmäisellä kerralla se vain laskee, mihin osoitteisiin mikäkin käsky sijoitetaan ja merkitsee omaan taulukkoonsa annetut nimet ja niitä vastaavat muistiosoitteet. Toisella kerralla assembler tietää eteenpäinkin tapahtuvista hyppyistä lopullisen osoitteen, joten se voi sijoittaa sen oikealle paikalleen.

Tämän kirjan lopussa on listattuna eräs assembler-ohjelma, MikroASSEMBLER, joka on jo varsin hyvä työkalu konekielisessä ohjelmoinnissa. Kirjan seuraavissa kappaleissa olevat esimerkit on suunniteltu siten, että voit kirjoittaa esimerkkiohjelmat ja ajaa ne käyttäen apuna tätä assembleria.

Ennen kuin jatkat kirjan lukemista tämän kappaleen jälkeen, olisi hyvä, että kirjoittaisit assembler-ohjelman koneeseesi ja tallentaisit sen kasetille tai levykkeelle, jotta se olisi käytettävissäsi, kun sitä tarvitset. Parhain tapa oppia jotain uutta on kokeilla käytännössä jokaista tiedon palasta erikseen. Suosittelen, että todella kirjoittaisit ja ajaisit jokaisen eteen tulevan ohjelmaesimerkin, ja tutkisit ohjelmien rakennetta.

Assembly-kielellä tässä kappaleessa esitetty A-kirjaimen kirjoitusohjelma olisi seuraavanlainen:

|              |            |               |                                        |
|--------------|------------|---------------|----------------------------------------|
| <b>\$850</b> | <b>LDA</b> | <b>#\$41</b>  | (lataa akkuun numero \$41)             |
| <b>\$852</b> | <b>STA</b> | <b>\$0400</b> | (vie akun sisältö osoitteeseen \$0400) |
| <b>\$855</b> | <b>RTS</b> |               | (palaa Basiciin)                       |

Kun syötät kirjan esimerkkejä MikroASSEMBLER-ohjelmalla koneeseesi, täytyy sinun aivan ensimmäiseksi kertoa ohjelmalle aloitusosoite, eli se osoite, josta ohjelman sijoittaminen muistiin aloitetaan. Commodore 64:ssä ja VIC-20:ssä voit käyttää alkuosoitteena \$033C:tä, mikäli et aio tallentaa esimerkkejä kasetille. Applen käyttäjät voivat aloittaa osoitteesta \$0300 ja Ataristit \$0600:sta. Pidemmille konekielisille ohjelmille on kuitenkin varattava tilaa jostain muualta. Kirjan konekohtaisissa luvuissa kerrotaan sopivien alueiden osoitteita.

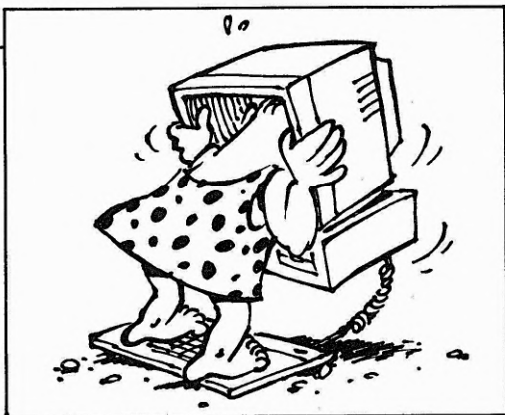
Kun käytät eri aloitusosoitetta kuin kirjan esimerkissä, muista myös muuttaa kaikki hyppyosoitteet omia osoitteitasi vastaaviksi.

Huomaat, että samoin kuin Basic-ohjelmissa voidaan ohjelman kulkua selventää REM-lauseilla, voidaan konekielessäkin lisätä assembler-listauksiin kommentteja. Tässä tekstissä kommentit on erotettu ohjelmasta sulkumerkeillä, mutta assembler-ohjelmat käyttävät erottimena puolipistettä (;) tai jotain muuta merkkiä ohjelmasta riippuen. MikroASSEMBLER ei kuitenkaan tallenna kommentteja, koska se on suoraan koodaava ohjelma.

Ohjelman viimeisenä käskynä on RTS, joka palauttaa ohjelman kontrollin pääohjelmalle. Koska konekieliohjelma kutsutaan Basicista käsin SYS-komennolla, joka vastaa konekielisen aliohjelman kutsua, palataan RTS:n jälkeen Basic-tulkkiin.

**Retki mikropiirimaahan**

# 4



## Kohti ydintä

Konekielinen ohjelmointi tunkeutuu suoraan laitteen syvyyksiin, alueille, jonne Basicilla ei ole asiaa. Kaikki aktiivinen toiminta tapahtuu yhden mikropiirin sisällä rekistereissä, jotka Basic-ohjelmoijalle ovat tuiki tuntemattomia. Myös Basic-ohjelmat suoritetaan täällä tietokoneen ytimessä, mutta kontrolli tapahtumien kulkuun on muualla; ei Basicistin käsissä.

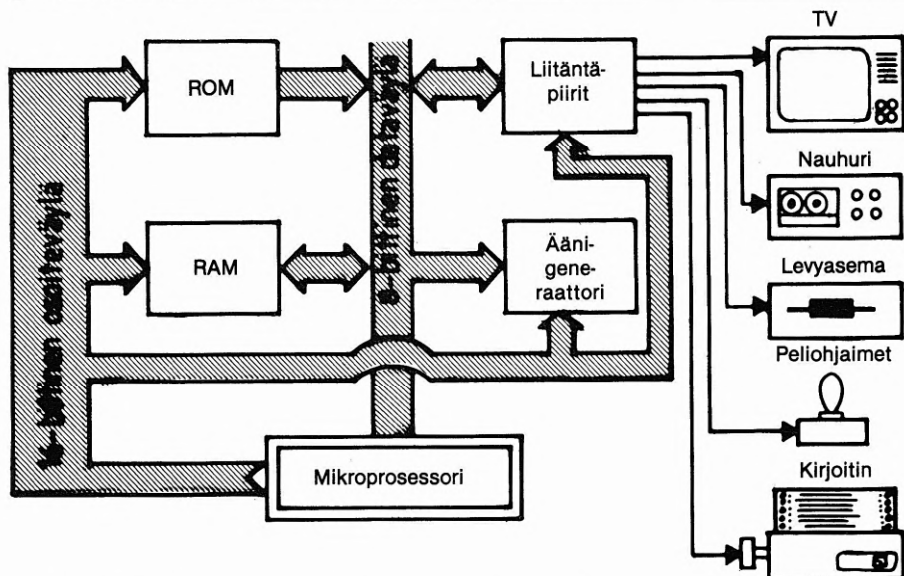
Oppiessasi konekielen pääset ohjaamaan koneen toimintaa parhaalla ja tehokkaimmalla tavalla, vaikuttamalla suoraan niihin pieniin rekisterityöläisiin, jotka koneen sisällä hoitavat käytännön työn. Konekieli on koneen oma kieli. Sillä puhut koneelle suoraan. Et tarvitse tulkkia välittämään sanomaasi.

Konekieli on nopea ja täsmällinen. Mutta ennen kuin pääset yhteenkään käskyyn kunnolla käsiksi, on sinun tunnettava alueet mikroprosessorin porttien takaa. Tietokoneen rakenteen tutkiminen on kuin murtautuisi kuorien läpi aivan uuteen maailmaan, joka koostuu rekistereistä, niissä asuvista biteistä, ja rekistereitä yhdistävistä valtateistä, väylistä.

### Tunne tietokoneesi

Olet mahdollisesti jo käsikirjastasi tutustunut oman tietokoneesi rakenteeseen. Tiedät sen koostuvan RAM-muistista, johon voidaan kirjoittaa tietoa ja josta voidaan lukea tietoa; ROM-muistista, josta voidaan vain lukea tietoa ja johon ei voi kirjoittaa. Tiedät myös, että muistin ohella laitteen sisällä on ääntä synnyttäviä piirejä, televisiolle tai monitorille kuvasignaaleja lähettäviä piirejä, joystickien eli peliohjainten liitännät ja kirjoitinliitäntä. Tiedät, että kaiken tämän toimintaa ohjaa yksi elektroninen piiri, mikroprosessori.

6502 on 8-bittinen mikroprosessori, mikä tarkoittaa, että tieto kulkee laitteen sisällä ja sitä käsitellään 8 rinnakkaisen johtimen välityksellä. Kukin johto vastaa



*Tietokoneen yleinen rakennekaavio. Osoite- ja dataväylään liitetään kaikki keskusyksikön ohjaamat piirit. 6502 keskusyksikkö näkee muistiavaruudessa vain peräkkäisiä muistipaikkoja. Oheislaitteet, kuten TV, levyasema tai kirjoitin, kytketään väyliin jonkin liitäntäpiirin kautta, jonka keskusyksikkö näkee tavallisena muistipaikkana. Kuvasta puuttuu eräs tärkeä oheislaitte, näppäimistö.*

yhtä kaksijärjestelmän numeroa, eli ykköstä tai nollaa. Rinnakkain ne pystyvät kuvaamaan lukuja välillä 0 — 255.

Mikroprosessorin hallinnassa on suuri määrä muistipaikkoja. Ne ovat elektronisia piirejä, jotka pystyvät säilyttämään niille annetun luvun muistisoluiissaan, joita on aina rinnakkain 8 kappaletta. Mikroprosessori on yhdistetty näihin muistipaikkoihin 8 rinnakkaisella johtimella, joita pitkin tieto siirtyy muistiin tai muistista mikroprosessoriin.

Tavalla tai toisella mikroprosessorin on kyettävä osoittamaan kutakin muistipaikkaa erikseen, jotta yksittäinen muistipaikka tietäisi, milloin juuri siihen halutaan tallentaa luku tai lukea sen sisältämä luku. Tämä on tietokoneessa järjestetty siten, että mikroprosessorilta kulkee jokaisen muistipaikan "takaovelle" vielä 16 sähköjohdinta. Kuten voit päätellä, jokainen niistä vastaa yhtä bittiä ja yhdessä ne muodostavat 16-bittisen binaariluvun. Kun 8 bitillä pystyttiin esittämään 256 erilaista lukua (0 — 255), kykenee 16 bittiä kuvaamaan 65536 eri lukua (0 — 65535).

Kullekin muistipaikalle on elektronisesti annettu ominaisuus, että ne pystyvät tunnistamaan yhden näistä luvuista. Kun jokaisella muistipaikalla on oma "tunnuslukunsa", meillä on suuri muistialue, johon voimme peräkkäin tallentaa

8-bittisiä lukuja laittamalla näille 16 johdolle peräkkäisiä binaarilukuja. Samoil-la binaarisilla "osoitteilla" voimme jälleen hakea samat luvut tarvittaessa takai-sin.

Koko muistialue on kuin pitkä katu (8-bittiiä leveä), jonka varrella sijaitsee muistipaikkoja. Mitä suurempi osoite, sitä kauempana kadun varrella sen osoit-tama muistipaikka sijaitsee.

Koko muistialue ei kuitenkaan ole varattu tällaisille muistipaikoille, joihin voi kirjoittaa ja joista tiedon voi myöhemmin hakea (RAM-muisti). Suurin osa muistista toki on tällaista ja niillä alueilla sijaitsevat mm. Basic-ohjelmasi.

Osa muistialueesta on kuitenkin varattu sellaisille muistipaikoille, joissa on pysyvästi tiettyjä lukuja (ROM-muisti). Ne paikat sisältävät perusohjelmat, jotka saavat tietokoneen käynnistymään, kun kytket siihen virran. RAM-muistissa on se huono ominaisuus, että sen sisältämä tieto katoaa välittömästi, kun virta kat-kaistaan. ROM-muistin tieto on sen sijaan poltettu siihen niin pysyvästi, että se säilyy vaikka laitteen virta olisi kytketty pois päältäkin.

Pienten perusohjelmien lisäksi ROM-muisti sisältää yhden varsin laajan ja tärkeän ohjelman, Basic-tulkin. Sen avulla tietokone aina ymmärtää antamasi Basic-käskyt ja pystyy RUN-käskyllä ajamaan ohjelmasi.

Yleensä Basic-tulkki muine kiinteine ohjelmineen ja taulukkotietoineen vie tietokoneen 64 kilotavun (65536) muistitilasta noin 20 kilotavua.

Jääkö RAM-muistille sitten loput 44 kilotavua? — Ei aivan. Mikroprosessorin on toki keskusteltava muidenkin kuin RAMin ja ROMin kanssa. Ohjelmiesi on pystyttävä ohjaamaan koneen äänigeneraattoria ja hakemaan tietoa peliohjaime-lta. Samaten tietoa on myös siirrettävä kirjoittimelle, levyasemalle ja muille oheis-laitteille.

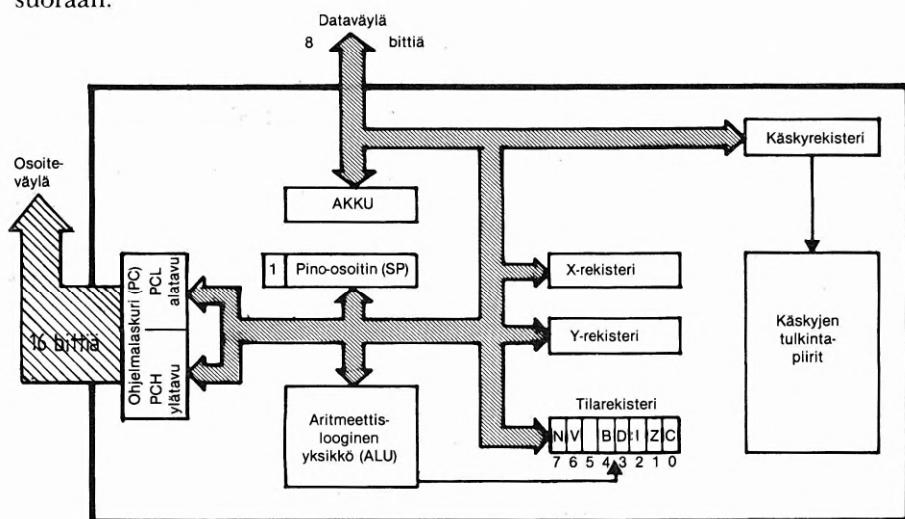
6502-prosessorissa nämä tiedonsiirrot on järjestetty siten, että jokin muisti-paikka tai useampikin toimivat ikäänkuin lännenelokuvista tutut kulissitalot. Mikroprosessori näkee ne aivan tavallisina muistipaikkoina, mutta rekisterin ta-kana onkin monimutkaisempaa elektroniikkaa, joka liittää niihin oheislaitteita. Koko laitteen toimintaa ohjaava keskusyksikkö luulee, että se siirtää tietoa vain samanlaiseen muistipaikkaan kuin muutkin ovat. Nämä osoitteet vievät hieman tilaa normaalilta RAM-muistilta ja ne ovat erittäin tärkeitä konekielessä, koska ne muodostavat kanavat ulkomaailmaan. Mitä iloa on ohjelmasta, jonka toimin-ta ja tulokset eivät näy missään?

## Pääportista sisään

Nyt alkaa läheinen kosketuksemme konekieleen. Käymme yksitellen läpi kaikki ne rekisterit 6502:n sisällä, jotka ovat käytössäsi ohjelmoidessasi konekielellä. Kaikkien toimintaan et pääse suoraan vaikuttamaan, mutta ne näyttelevät kui-tenkin kaikki varsin keskeistä osaa konekielisissä ohjelmissasi.

Tämä alue tietokoneesta on täysin Basicilta suljettua. Millään keinolla Basic-

ohjelmoija ei voi tutkia eikä muutella tai ohjata näiden rekistereiden sisältöä suoraan.



Keskusyksikön sisällä on useita rekistereitä, 8-bittisiä muistipaikkoja. Niillä ei ole samanlaista osoitetta kuin ulkoisilla muistipaikoilla, vaan niiden käsittelyyn on omat konekielen käskynsä. Akku on rekistereistä tärkein. Sen kautta kulkee useimmat käsiteltävät tiedot. Selvyyden vuoksi osoiteväylä on kytketty vain 16-bittiseen osoitelaskuriin. Käytännössä osoiteväylälle voi lukuja siirtyä myös suoraan muilta rekistereiltä ja sisäiseltä väylältä.

## Kuusi rekisteriä

Salattu maailmamme koostuu kuudesta rekisteristä. Oikeastaan rekistereitä on enemmänkin, mutta ainoastaan kuusi on näkyvissä konekieli-ohjelmoijalle. Muut ovat mikroprosessorin suunnitteluvaiheessa sinne lisättyjä ja ne osallistuvat ainoastaan käskyjen toteuttamisessa avustamiseen. Niiden käsittelemiseen ei ole mitään keinoa.

Kaikki kuusi meitä ensi sijassa kiinnostavaa rekisteriä ovat kiinni samassa 8-bittisessä väylässä kuin mikroprosessorin ulkopuoliset muistipaikatkin. Itse asiassa nämä rekisterit ovat myös samankaltaisia muistipaikkoja. Niille vain ei ole olemassa osoitteita vaan niitä ohjataan suoraan konekielen käskyillä. Näille rekistereille on elektronisilla kytkennöillä saatu aikaan ominaisuuksia, joita ei tavallisilla muistipaikoilla ole ja jotka tekevät mahdolliseksi koko tietokoneen toiminnan. Muistista voidaan siirtää niihin lukuja ja päinvastoin: rekisterien sisältö voidaan siirtää haluttuun muistipaikkaan.

Kuten kaikki siirrot tietokoneen muistipaikkojen välillä, on tässäkin kyseessä



kopiointi. Rekisterissä säilyy edelleen alkuperäinen bittitieto vaikka sen sisältö siirretäänkin, kopioidaan, toiseen paikkaan.

Meitä kiinnostavat rekisterit ovat: akku, X-rekisteri, Y-rekisteri, ohjelmanaskuri, pino-osoitin ja tilarekisteri. Näkymättömiä apurekistereitä ovat käskyrekisteri ja laskentaa suorittavan aritmeettis-loogisen yksikön (ALU) sisältämät rekisterit.

## Akku on kiireinen

Akku on ehdottomasti tärkein kaikista rekistereistä. Sitä käytetään runsaimmin. Sen kautta siirtyy yleensä tiedot muistiin ja takaisin ja sillä on ennen kaikkea laskentaan liittyviä ominaisuuksia, joita ei muilla rekistereillä ole. Akussa lasketaan 8-bittisiä lukuja yhteen ja vähennetään sekä suoritetaan vertailuja lukujen kesken. Nimitys 'akku' tulee sen englanninkielisestä nimestä 'accumulator', joka tarkoittaa 'varastoija'.

Heksadesimaalilukuna ilmaistu konekielen käsky, joka siirtää luvun akkuun on \$A9 (binaarisena 10101001, 10-järjestelmässä 169). Tämä käsky tarkoittaa, että välittömästi seuraavassa tavussa on luku, joka halutaan siirtää akkuun. Esimerkiksi \$A9 \$5F sijoittaa akkuun heksaluvun 5F (binaarisena 01011111). Tämän käskyn selväkielisempi merkintätapa, jollaisen konekieliohjelmoinnissa apuna käytettävät assembler-ohjelmat hyväksyvät on

**LDA #\$5F (LoaD Accumulator, lataa akkuun luku \$5F)**

Tästä eteenpäin esitämme käskyt aina assemblerin yhteydessä käytettävänä kirjainlyhenteinä. Ne on huomattavasti helpompi muistaa kuin pelkät numerosarjat ja toisaalta konekielistä ohjelmointia harrastetaan pääsääntöisesti assemblerin avulla, jolloin on syytä tottua sen käyttöön.

Hyvin lyhyitä konekielisiä ohjelmanpätkiä syötetään koneeseen usein pelkän monitorin avulla tai Basicin DATA-lauseisiin liitettynä. Silloin ei ole assemblerin tukea, mutta lyhyiden pätkien koodaaminen käsin ei ole hankalaa. Kirjan lopussa olevasta taulukosta katsotaan kunkin käskyn koodi heksalukuina (monitorilla syötettäessä) tai desimaalilukuina (DATA-lauseisiin sijoitettaessa).

LDA-käskyä on olemassa 8 eri muotoa. Edellä esitetty \$A9 on yksi niistä ja se sijoittaa akkuun välittömästi seuraavassa tavussa (eli itse ohjelman sisällä) olevan luvun. Akkuun voi siirtää luvun myös mistä tahansa muistipaikasta RAM-alueelta, esimerkiksi käskyllä

**LDA \$2000,**

joka lataa akkuun muistipaikan \$2000 sisällön. Tällä tavalla akkuun ladattaessa käskyn heksamuoto on \$AD. Mikroprosessorille nämä ovat kaksi eri käskyä ja sen vuoksi se osaa \$A9:n kohdalla sijoittaa akkuun välittömästi seuraavan tavun itse ohjelmassa ja \$AD:n kohdalla se osaa hakea luvun paikasta, joka löytyy, kun kahta seuraavaa tavua käytetään osoitteena ja käydään se sieltä hakemassa.

Assembly-kielellä ohjelmoitaessa käskyn merkintätapa on kuitenkin sama:



LDA. Assembler-ohjelmalle on siis muulla tavoin ilmaistava, mikä latauskäsky on kyseessä. Se tapahtuu tietyillä merkeillä osoitetta tai lukua kuvaavan numero-sarjan edessä.

Ensimmäisen LDA:n yhteydessä (\$A9) lukua \$5F edelsi # -merkki eli 'risuaita'. Siitä assembler ymmärtää, että luku \$5F on itsessään akkuun sijoitettava luku ja osaa sijoittaa ohjelmaan käskyn \$A9. Toisessa esimerkissä ei ollut risuaitaa, ainoastaan \$ osoittamassa heksalukua. Tästä assembler ymmärtää, että kyseinen heksaluku tarkoittaa osoitetta, josta ladattava luku on haettava ja osaa sijoittaa ohjelmaan käskyn \$AD. Itse mikroprosessorille käskyt ovat kuitenkin erilaisia eikä sekaannuksia synny.

6502:lla on sellainen omaperäinen tapa, että se haluaa osoitteesta vähemmän merkitsevät numerot ensin. Kun osoitteemme, josta akkuun sijoitettava luku haetaan, oli \$2000, niin mikroprosessori vaatii, että käsky löytyy muistista muodossa: \$AD \$00 \$20. Kaikki nämä ovat peräkkäisissä muistipaikoissa siellä, missä mikroprosessori käy ohjelmaa läpi.

Assembler-ohjelmamme huolehtii tästä osoitetavujen kääntämisestä automaattisesti, eikä meidän tarvitse siihen puuttua. Käsien koodattaessa se kuitenkin on muistettava: osoitteen tavut käänteisessä järjestyksessä!

Nämä kaksi tapaa ladata luku akkuun kuvaavat erilaisia 'osoitusmuotoja'. Osoitusmuodot ovat tapoja, joilla käskyn yhteydessä kerrotaan, mistä osoitteesta haettu tieto löytyy — tai pikemminkin, miten tämä osoite muodostetaan.

Osoitusmuodoista on oma erillinen lukunsa, koska niiden hallitseminen on keskeisimpiä asioita konekielen käytössä. Siinä luvussa selvitetään myös tarkemmin, miten eri osoitusmuodot ilmaistaan assemblerin yhteydessä, joten emme tässä vaiheessa vaivaa sillä enempää päättämme. Osoitusmuotoja on muuten yli kymmenen ja ne mahdollistavat varsin monipuolisen muistin hallinnan.

Akusta tieto siirtyy muistiin käskyn STA avulla (STA: STore Accumulator = tallenna akku). Myös STA-käskystä on olemassa eri versioita, kuten LDA:stakin. Välitöntä osoitusmuotoa ei kuitenkaan ole käytettävissä, eli välittömästi käskyn perässä olevaa tavua ei voi tallentaa muistiin. Tämä on varsin ymmärrettävää, sillä käskyhän taltioi nimenomaan AKUN sisällön, eikä siis mitään muuta.

Muuten STA-käsky toimii samalla tavalla kuin LDA:kin. STA \$2000 tallettaisi akun sisällön muistipaikkaan \$2000 (muista, että \$ tarkoittaa heksalukua). Se, mitä muistipaikassa sattui olemaan sillä hetkellä, häviää ja tilalle tulee akun sisältämä luku.

On siis oltava varovainen, ettei tuhoa muistista mitään oleellista — aivan kuin Basicin POKE-käskyn suhteen on oltava. Vääriin paikkoihin POKEileminen saattaa johtaa koneen sekoamiseen.

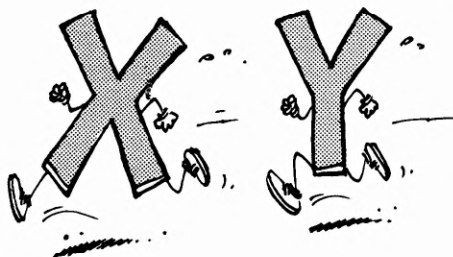
LDA ja STA muistuttavat monessa suhteessa Basicin käskyjä PEEK ja POKE. PEEKillä haetaan muistipaikasta luku ja POKElla sinne kirjoitetaan jokin luku. Ainoa ero on, että Basicissa voimme PEEKillä hakea luvun ja sijoittaa sen jonkin

muuttujan arvoksi. POKella voimme sijoittaa muistiin minkä tahansa luvun, vaikkapa muuttuja arvon. Konekielessä LDA ja STA sen sijaan käsittelevät ainoastaan akun sisältöä. Akku on konekielen muuttuja.

Jos tutkit konekielisen ohjelman rakennetta, esimerkiksi oman laitteesi Basic-tulkkia, joka on itse asiassa laaja konekielinen ohjelma, huomaat LDA- ja STA-käskyjen olevan ylivoimaisesti suosituimpia. Se johtuu siitä, että käsitelläkseen mitä tahansa lukua millä tavalla tahansa, on mikroprosessorin yleensä ensin haettava se muistista ja sen jälkeen tallennettava se sinne takaisin vapauttaakseen tilaa uusille luvuille.

Konekielistä ohjelmaa tutkitaan joko yksinkertaisen monitorin avulla, joka antaa ohjelman sinulle heksalukuina tai disassemblerin avulla, joka osaa kääntää konekielen numerokäskyt selväkieliseksi kirjainlyhenteiksi, kuten LDA ja STA. MikroASSEMBLER sisältää myös disassembler-toiminnan.

## Juoksupojat X ja Y



Akun tukena lukujen hakemisessa ja tallentamisessa toimii kaksi niinikään 8-bittistä rekisteriä. Niitä kutsutaan X- ja Y-rekistereiksi. Ne eivät pysty samanlaisiin matemaattisiin suorituksiin kuin akku. Oikeastaan ne osaavat vain lisätä sisältämäänsä lukuun yhden tai vähentää siitä yhden.

X- ja Y-rekistereitä käytetään pääasiassa osoitteen yhteydessä muutamissa osoitusmuodoissa. Esimerkiksi seuraava käsky:

**LDA \$2000,X**

ei lataakaan akkuun lukua osoitteesta \$2000, vaan osoitteesta \$2000 + X. Osoitteeseen siis lisätään X-rekisterin sisältämä luku. Mitä hyötyä tästä on? Emmekö olisi voineet lisätä sitä lukua jo ohjelmaa tehdessämme? Olisimme toki, mutta jippo piilee siinä, että voimme muuttaa X:n arvoa ja sen jälkeen esimerkiksi käskymme hakisikin luvun akkuun jo eri paikasta.

Tämä yhdistettynä X-rekisterin kykyyn inkrementoida (lisätä yhdellä) ja dekrementoida (vähentää yhdellä) sisältämäänsä lukua antaa meille oivat eväät käsitellä konekielellä useita peräkkäisiä muistipaikkoja: käytämme käskyä LDA \$2000,X — lisäämme X-rekisterin arvoa yhdellä (käskyllä INX) ja suoritamme LDA-käskyn uudestaan. Näin voimme vaivattomasti 'liukua' pitkin muistiavaaruutta.

Esimerkiksi seuraava konekielinen ohjelmanpätkä siirtää kuvaruudulle tekstin

"SYNTAX ERROR". Ohjelma olettaa, että muistipaikkoihin osoitteesta \$0370 (880) lähtien on sijoitettu tekstin muodostavat kirjainkoodit. Commodore 64:ssä sinne on ennen SYS 848 (tai .G848) komentoa POKETTAVA luvut 19, 25, 14, 20, 1, 24, 32, 5, 18, 18, 15 ja 18. Ne voi sijoittaa muistiin myös MikroASSEMBLERin monitorilla.

- \$0350 LDX #\$00** (aseta X-rekisteriin nolla)
- \$0352 LDA \$0370,X** (hae akkuun luku osoitteesta \$0370 + X. Ensimmäisellä kerralla osoite on \$0370, toisella kerralla \$0371 jne.)
- \$0355 STA \$0400,X** (vie luku saman tien kuvaruutumuistiin, jossa se sytyttää ruudulle koodilukunsa mukaisen kirjaimen. Ensimmäisellä kerralla luku viedään kuvaruutumuistin ensimmäiseen tavuun, X = 0. Toisella kerralla luku viedään toiseen tavuun, X = 1 jne.)
- \$0358 INX** (lisätään X-rekisterin sisältöön yksi, jotta siirtyisimme osoittamaan sanomamme seuraavaa kirjainta ja vastaavasti siirtyisimme kuvaruutumuistissa yhden pykälän eteenpäin)
- \$0359 CPX #\$0C** (CPX on käsky, joka vertaa X-rekisterin sisältämää lukua tässä tapauksessa heksalukuun \$0C (12), joka on sanomamme pituus. Käsky ei muuta X-rekisterin sisältöä)
- \$035B BNE \$0352** (tämä käsky käyttää hyväkseen edellisen käskyn vertailun tulosta: jos X-rekisterin sisältö ei ollut kasvanut vielä \$0C:hen, HYPPÄÄMME ohjelmassa takaisin muistipaikkaan \$0352 suorittamaan siellä olevaa käskyä uudestaan (uudella X:n arvolla). Jos taas X-rekisteri oli jo kasvanut lukuun \$0C, emme hyppää vaan jatkamme normaalisti seuraavasta käskystä)
- \$035D RTS** (RTS-käsky siirtää meidät takaisin pääohjelmaamme tästä pienestä aliohjelmasta. Jos olimme käynnistäneet sen Basicista SYS-käskyllä, palaamme takaisin Basiciin)

Tällä tavalla X-rekisterin (ja vastaavasti Y-rekisterin) avulla 'juoksutetaan' osoitetta pitkin jotain muistialuetta ilman, että meidän tarvitsee kirjoittaa ohjelmaamme samaa käskyä erikseen jokaiselle haluamallemme osoitteelle.

Kuvittele, kuinka pitkiä ohjelmistasi tulisi, jos joutuisit kirjoittamaan kaikki käskyt erikseen siirtäessäsi esimerkiksi 1000 tavua paikasta toiseen (yksi kuvaruudullinen tekstiä)! Käyttämällä haarautumiskäskyä suoritamme saman ohjelma-kohdan toistuvasti peräkkäin erilaisilla X- (tai Y-) rekisterin arvoilla.

Basicissa on paljolti tätä muistuttava käskyrakenne: FOR...NEXT. Edellinen ohjelma olisi voitu tehdä Basicilla seuraavasti:

|                      |                                                                                                                                                                                          |
|----------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 10 FOR I = 0 TO 11   | (aseta muuttuja I liukumaan 0:sta 11:een)                                                                                                                                                |
| 20 A = PEEK(880 + I) | (hae muuttujaan A luku muistipaikasta 880 lisättynä muuttujan I arvolla. Muista: PEEKin ja POKEn yhteydessä käytetään 10-järjestelmän lukuja! \$0370 = 880)                              |
| 30 POKE 1024 + I,A   | (sijoita luku muistipaikkaan 1024 lisättynä jälleen muuttujan I arvolla)                                                                                                                 |
| 40 NEXT I            | (kasvata I:n arvoa yhdellä. Basic testaa samalla, onko muuttuja jo saavuttanut FOR-käskyllä annetun tavoitearvonsa. Jos on, jatketaan ohjelmaa eteenpäin ilman takaisinpäin siirtymistä) |
| 50 RETURN            | (siirrytään takaisin pääohjelmaan. Vastaa täysin konekielen käskyä RTS)                                                                                                                  |

X- ja Y-rekisterien käyttö osoitteen yhteydessä tekee konekielessä mahdolliseksi helpon taulukoiden käsittelyn. Näitä rekistereitä käytetään taulukon indekseinä ja usein X-ja Y-rekistereitä kutsutaankin indeksirekistereiksi.

## Ohjelmalaskuri (PC, Program Counter)) ja käskyrekisteri

Olet varmaan ihmetellyt, miten mikroprosessori pystyy lukemaan käskyjä muistista. Eihän se voi muistiin nähdäkään, joten tavalla tai toisella sen on ne käskyt sieltä haettava pystyäkseen tulkitsemaan ohjelmaa.

Näitä toimintoja varten mikroprosessorissa on kaksi rekisteriä: ohjelmalaskuri, jota merkitään lyhenteenä PC (Program Counter), ja käskyrekisteri.

Ohjelmalaskuri on kahden tavun levyinen rekisteri, eli siis 16-bittinen. 16 bitillä voidaan esittää mikä luku tahansa väliltä 0 — 65535, joten jos sitä käytetään osoitteena, voidaan osoittaa mitä muistipaikkaa tahansa koko käytettävissä olevalta 64 kilotavun muistialueelta.

Ohjelmalaskuri on rekisteri, joka aina sisältää seuraavaksi suoritettavan käskyn osoitteen. Kun Basicista käsin annat esimerkiksi käskyn

### **SYS 3000**

sijoittaa Basic-tulkki alkutoimien jälkeen ohjelmalaskuriin antamasi luvun 3000 (heksana \$0BB8) ja seuraava käsky, jonka keskusyksikkö hakee, onkin sinun konekieliohjelmasi ensimmäinen käsky (mikäli olit tallettanut ohjelmasi muistiin alkaen osoitteesta \$0BB8).

Ohjelmalaskurin sisältämä luku siis menee 16 osoitejohdinta (osoiteväylä) pitkin muistille, jossa kyseinen muistipaikka kuuliaisesti antaa sisältämänsä luvun 8-bittistä dataväylää pitkin keskusyksikölle.

Keskusyksikön sisällä tätä käskyä ei laiteta mihinkään meidän käytössämme olevaan rekisteriin, vaan se sijoitetaan erityiseen käskyrekisteriin, jossa sitä säilytetään seuraavan käskyn hakuun asti. Keskusyksikön monimutkaiset elektroniset

piirit tulkitsevat käskyn merkityksen ja suorittavat sen vaatimat toimenpiteet (esimerkiksi lisäävät X-rekisterin arvoa yhdellä jne).

Usein käskyn toteuttamiseksi on noudettava vielä pari seuraavaakin tavua ohjelmasta, koska ne saattavat sisältää esimerkiksi osoitetiedon LDA-käskylle. Keskusyksikön ohjauspiirit hakevat nämä välttämättömät käskyyn liittyvät argumentit kasvattamalla ensin ohjelmalaskurin sisältöä yhdellä ja tarvittaessa vielä toisella.

Näitä osoitetietoja käytetään sitten hyväksi itse käskyn toteuttamisessa: ladataan akkuun luku tai taltioidaan akun sisältö. Kun viimeinenkin argumentti on haettu, kasvatetaan ohjelmalaskurin sisältöä vielä yhdellä, jotta se osoittaisi jälleen seuraavaksi suoritettavaa käskyä.

On erittäin tärkeää huomata, että konekielessä jokainen muistissa oleva tavu ei ole konekielen käsky. Itse käsky (LDA, STA jne.) on kylläkin yhden tavun mittainen, mutta sitä seuraa usein osoitetietoja, jotka kuuluvat tähän samaan käskyyn. Kullakin käskyllä (ja osoitusmuodolla) on aivan oma koodinsa, josta keskusyksikkö tietää, kuinka monta tavua kuuluu samaan käskyyn.

Tästä johtuu, ettet voi siirtyä suorittamaan konekieliohjelmaa mistä kohtaa tahansa. Todennäköisyys on suuri sille, että ensimmäiseksi 'käskyksi' tulisikin osoitetieto, joka luonnollisesti ei voi olla keskusyksikölle mikään järkevä käskykoodi. Kone todennäköisesti joko sekoaisi tai lähtisi toimimaan aivan väärin.

Käskyrekisteri on siis eräs niistä piilorekistereistä, joita ei konekielisessäkään ohjelmoinnissa pystytä käsittelemään. Se toimii orjamaaisena apurekisterinä, jonka olemassaolosta ei tarvitse lainkaan välittää. On hyvä kuitenkin tuntea myös tämä puoli keskusyksikön toiminnasta, jotta voisi rauhassa keskittyä itse ohjelmointiin. Käytännön ohjelmointityössä voimme kuvitella, että keskusyksikkö 'näkee' muistissa olevan ohjelman ohjelmalaskurinsa avulla eikä hae käskyjä tutkittavakseen.

Ohjelmalaskurinkaan sisältöön emme voi suoranaisesti puuttua. Keskusyksikkö hoitaa taloudenpidon myös ohjelmalaskurin suhteen täysin. Meidän tarvitsee ainoastaan tietää, että se osoittaa aina seuraavaksi suoritettavaa käskyä ja meidän on seurattava sen arvon muuttumista, jotta voimme ohjelmassamme hypätä tai haarautua oikeisiin paikkoihin.

Emme voi ladata ohjelmalaskuriin lukua samalla tavalla kuin esimerkiksi akkuun tai X-rekisteriin. Voimme kuitenkin antaa käskyn JMP (JuMP = hyppää). Se vastaa Basicin käskyä GOTO. Esimerkiksi

### **JMP \$2010**

aiheuttaa sen, että ohjelmalaskurin arvoksi tulee \$2010 ja seuraavaksi suoritettava käsky onkin juuri \$2010:ssa sijaitseva käsky eikä JMP-käskyä seuraavassa tavussa oleva käsky.

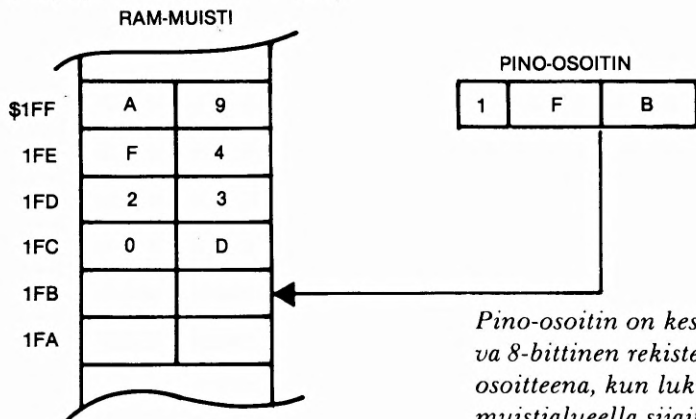
## Osoitteet pinoon



Pino-osoitin (SP: Stack Pointer) on varsin vekkuli osa keskusyksikköä. Siihen ei voi tallentaa tietoa, mutta se osallistuu oleellisella tavalla keskusyksikön toimintaan aliohjelmien suorittamisessa. Se tallentaa aliohjelmaan hypättäessä paluusoitteen.

Kun aliohjelma on suoritettu, halutaan luonnollisesti palata takaisin samaan paikkaan, tai tarkemmin ottaen seuraavaan käskyyn, kuin mistä aliohjelmaan lähdettiin. Pino-osoitin hoitaa siis paluusoitteiden arkistoinnin.

Pino-osoitin on nimensä mukaisesti vain osoite. Se osoittaa tietylle alueelle muistiavaruudessa, ns. toiselle muistisivulle. Yksi sivu on aina 256 tavua, joten pino-osoitin, joka on 8 bittiä leveä sekkin, voi osoittaa osoitteita \$0100 — \$01FF (10-järjestelmässä 256 — 511).



*Pino-osoitin on keskusyksikössä sijaitseva 8-bittinen rekisteri. Sitä käytetään osoitteena, kun lukuja tallennetaan muistialueella sijaitsevaan pinomuistiin tai otetaan sieltä takaisin.*

8-bittinen rekisteri voisi osoittaa sellaisenaan vain osoitteita 0 — 255, mutta pino-osoittimen osoitusalue on 'nostettu' seuraavalle sivulle lukitsemalla osoiteväylässä 9. bitti pysyvästi ykköseksi pino-osoittimen antaessa sinne tietonsa.

Kun tietokoneeseen kytketään virrat päälle, se sijoittaa pino-osoittimeen luvun \$FF. Koska 9. bitti on aina ykkönen pino-osoittimen osoittaessa muistia, on pino-osoittimen todellinen sisältö siis \$01FF eli 511. Se on pinon yläpää ja pinon alapää on osoitteessa \$0100 eli 256. Itse pino sijaitsee siis muistialucella ja pino-

osoitin on vain osoiterekisteri, joka osoittaa meitä kiinnostavaan kohtaan pinossa.

Kun keskusyksikkö konekieliohjelmaa toteuttaessaan törmää aliohjelmakutsuihin JSR (Jump to SubRoutine = hyppää aliohjelmaan), tapahtuu seuraavaa. Sen hetkinen ohjelmalaskurin arvo (joka JSR-käskyn hakemisen jälkeen osoittaa jo seuraavaan käskyyn) tallennetaan muistiin pino-osoittimen osoittamaan paikkaan. Ohjelmalaskurin merkitsevämpi (ylempi) tavu tallennetaan juuri siihen osoitteeseen, joka pino-osoittimessa oli. Sen jälkeen pino-osoittimen sisältöä vähennetään yhdellä ja ohjelmalaskurin vähemmän merkitsevä tavu tallennetaan tähän uuteen pino-osoittimen sisältämään osoitteeseen.

Tämän jälkeen pino-osoittimen sisältöä vähennetään jälleen yhdellä, jotta se olisi valmis suorittamaan saman operaation seuraavan JSR-käskyn tullessa. Niitä voi tulla (ja usein tulee) monta peräkkäin ilman, että välillä palataan pääohjelmaan. Basicissakin käytetään usein sisäkkäisiä aliohjelmakutsuja (GOSUB).

RTS (ReTurn from Subroutine = palaa aliohjelmasta) on käsky, joka kertoo keskusyksikölle, että aliohjelma on päättynyt ja on palattava samaan paikkaan, josta aliohjelmaa kutsuttiin. Juuri sen aliohjelman paluuosoite on viimeiseksi tallennettu pinoon (ei ole mielekästä, että paluukäsky viittaisi johonkin muuhun aliohjelmaan kuin juuri viimeksi kutsuttuun).

Paluu aliohjelmaa kutsuneeseen kohtaan tapahtuu hakemalla ohjelmalaskurin alempi tavu pino-osoittimen osoitteesta, johon on lisätty yksi (pino-osoitinnan osoitti seuraavaan tyhjään kohtaan) ja enemmän merkitsevä tavu pino-osoittimen osoitteesta lisättyinä kahdella. Samalla pino-osoittimen arvoa kasvatetaan kahdella, jotta se osoittaisi ensimmäiseen tyhjään paikkaan näin vapautuneessa tilassa.

Pino-osoitin siis liukuu yli sen hallitseman muistialueen sen mukaan, miten ohjelmassa kutsutaan aliohjelmia ja palataan niistä. Pino-osoitin osoittaa valmiiksi seuraavaa vapaata paikkaa ja kun halutaan palata pääohjelmaan pinosta vain haetaan viimeiseksi tallennettu paluuosoite. Tallennettaessa paluuosoitetta totesimme sen jo osoittavan pääohjelmassa seuraavaksi suoritettavaa käskyä, joten muuta ei tarvitse tehdä kuin siirtää paluuosoite ohjelmalaskuriin. Seuraavaa käskyä hakiessaan keskusyksikkö jo tottelee tätä uutta osoitetta.

Aliohjelmakutsu eli JSR-käsky sisältää myös aina osoitteen, josta aliohjelma löytyy. Käsky voi olla kokonaisuudessaan esimerkiksi

**JSR \$4500**

Tämä käsky siis siirtää ohjelmalaskurin sen hetkisen sisällön talteen pinoon. Jotta siirryttäisiin käskyn mukaan aliohjelmaan, ei tarvitse muuta kuin sijoittaa käskyn argumentti \$4500 ohjelmalaskuriin ja seuraavaa käskyä haettaessa jo ollaankin aliohjelmassa.

Pinoa voi käyttää muidenkin lukujen kuin paluuosoitteiden taltiointiin. Kes-



**kusyksikkö** tuntee käskyt PHA ja PLA, joilla vastaavasti siirretään akun sisältö **pinoon** (PHA = PusH Accumulator) ja pinosta takaisin akkuun (PLA = PuLl Accumulator). Samat toimenpiteet voidaan kohdistaa myös tilarekisteriin, jonka toimintaa tutkitaan seuraavassa kappaleessa.

Näiden käskyjen kanssa leikkimistä ei voi kuitenkaan suositella. Muistissa on riittävästi tilaa välitulosten tallentamiseen, joten siihen ei tarvitse välttämättä käyttää pinoa. Pino on lisäksi rajatun pituinen, joten jos käytät ohjelmassasi runsaasti lukujen pinotallennuksia, saattaa tila loppua aliohjelmien paluusoitteilta. Yhdenkin osoitteen katoaminen sekoittaa ohjelman kuin ohjelman täydellisesti.

Pinon tallentaminen on tosin nopeampaa kuin täydellä osoitteella muistiin vieminen ja se on lyhyempi, vain yhden tavun mittainen käsky. Jos joskus ohjelmassasi on erittäin aikakriittinen osuus, jonka suoritusaika on saatava mahdollisimman lyhyeksi, saattaa pinon tallentaminen osoittautua korvaamattomaksi apuneuvoksi. Joka tapauksessa varoitus on tässä kohdassa paikallaan: älä käytä pinoa turhaan, jätä se koneen huoleksi ja paluusoitteiden käyttöön.

Jos kuitenkin päädyt pinon käyttöön, muista, että **JOKAISTA PINOON PANO ON SEURATTAVA PINOSTA OTTO ENNEN RTS-KÄSKYÄ!** Jos pinon jää ylimääräistä "roskaa" ei paluusoitteemme enää olekaan pinossa päällimmäisenä ja ohjelmamme palaa ties minne paikkaan muistissa ja ohjelma kaahtuu varmasti.

## Liput salkoon

Viimeisin ohjelmoijan käytössä olevista rekistereistä keskusyksikön sisällä on niistä tärkein mielekkäiden ohjelmien tekemisen kannalta: tilarekisteri (SR: Status Register).

Tilarekisterin avulla keskusyksikkö pystyy tekemään ohjelman sisällä Basicista tuttuja IF..THEN -tyyppisiä päättelyjä ja haarautumaan ohjelmassa esimerkiksi silmukoiden muodostamiseksi.

Tilarekisteri on 8 bittiä leveä (joista bitti 5 on käyttämätön ja jatkuvasti ykköstilassa), mutta sen sisältöä ei keskusyksikkö eikä ohjelmoija ajattele lukuna vaan seitsemänä erillisenä muistilappuna. Se on kuin seitsemän rinnakkaista sähkölamppua, jotka voivat olla joko sammuneena tai syttyneenä. Niitä kutsutaan myös lipuiksi, koska ne liputtavat ohjelmallesi tärkeitä viestejä.

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| N | V | 1 | B | D | I | Z | C |
|---|---|---|---|---|---|---|---|

Bitti                      7        6        5        4        3        2        1        0

*Tilarekisteri on 8 bittiä leveä rekisteri, jonka 6. bitti eli bitti5 ei ole käytössä (aina 1). Tilarekisteriä ei käsitellä yhtenä lukuna vaan sen jokainen bitti on ikäänkuin lippu, joka voi liehua pystyssä (1) tai olla 'kaadettuna' (0).*



Tilarekisterin liput ovat N, Z, C, I, D ja V. Tavallisesti niistä käytetään vain Z- ja C-lippuja. Muita käytetään vain erittäin harvoin tai ei juuri ollenkaan.

Z (Zero = nolla) -lippu asettuu ykköseksi aina, kun käskyn kohteena olleen rekisterin sisältö muuttuu nolaksi. Esimerkiksi, jos keskusyksikkö suorittaa käskyn LDA #\$00 eli lataa akkuun nollan, asettuu Z-lippu ykköseksi oli se ennen käskyn suorittamista mitä tahansa.

Jos lataamme akkuun suoraan nollan on Z-lippuun katsomattakin itsestään selvää, että akun sisältö muuttui nolaksi. Jos sen sijaan lataamme akkuun luvun jostain muistipaikasta, jossa pidämme ohjelmamme muuttuvia tietoja, antaa Z-lippu mahdollisuuden tarkistaa sisältääkö tämä muistipaikka nolasta eroavan luvun.

Z-lippua tarkastellaan kahdella käskyllä: BEQ ja BNE. BEQ (Branch if EQual) testaa Z-lipun sisällön ja haarautuu käskyn yhteydessä annettavaan osoitteeseen, mikäli Z-lippu on ykkönen, eli jos edellä suoritettussa käskyssä akun (tai jonkin muun rekisterin) sisältö tuli nolaksi.

BNE (Branch if Not Equal) taas suorittaa haarautumisen, mikäli Z-lippu ei ollut ykkönen.

C-lippu tarkoittaa muistinumeroa. Se toimii aivan samalla tavalla kuin laskettaessa paperilla ja kynällä yhteen lukuja. Olettakaamme, että suoritamme seuraavanlaisen yhteenlaskun paperilla:

$$\begin{array}{r} \phantom{0}9 \\ 18 \\ \hline ? \end{array} \qquad \begin{array}{r} \phantom{0}9 \\ 18 \\ \hline 27 \end{array}$$

Laskimme siis yhteen luvut 9 ja 18. Niiden summa ensimmäisessä sarakkeessa on 17, mutta koska se ei mahdu sellaisenaan ykkösten sarakkeeseen, merkkääme kyseiseen paikkaan vain seiskan ja seuraavan sarakkeen päälle pienen ykkösen muistinumeroksi, että kymmeniä tuli edellisestä sarakkeesta lahjana yksi kappaletta.

Kahta numeroa yhteen laskettaessa ei muistinumero voi koskaan olla suurempi kuin yksi. Samalla tavalla laskiessamme lukuja yhteen keskusyksikön akussa, saattaa tulos ylittää akun kapasiteetin (joka on 255). Tässäkään tapauksessa muistinumero laskettaessa kahta lukua yhteen ei voi ylittää ykköstä, joten laskettaessa akussa yhteen lukuja keskusyksikkö voi sijoittaa muistinumeron 1-bittiin C-lippuun (C: Carry = muistinumero).

Yhteenlasku akussa tapahtuu käskyllä ADC (ADD with Carry = lisää akkuun muistinumeron kanssa). Esimerkiksi käsky ADC #\$05 lisäisi akkuun heksaluvun \$05. Muistathan, että risuaidan merkki (#) tarkoitti, että seuraava luku on itse käsiteltävä luku.

Käsky ADC \$200F lisäisi akkuun luvun, joka löytyy osoitteesta \$200F. Lisäys

tapahtuu muistinumeron kanssa sen takia, että joskus joudutaan käsittelemään pidempiä lukuja kuin yhteen tavuun mahtuu. Silloin luku sijaitsee useammassa tavussa ja yhteenlaskettaessa lasketaan yhteen ensin vähemmän merkitsevät tavut ja sen jälkeen enemmän merkitsevät tavut PLUS edellisestä laskutoimituksesta mahdollisesti jäänyt muistinumero.

Ensimmäistä yhteenlaskua täytyy aina edeltää käsky, jolla muistinumero nolataan, koska se saattaa jonkin edellisen operaation tuloksena olla asettunut ykköseksi.

C-lippua käytetään siis hyväksi yhteenlaskettaessa, mutta sen perusteella voi myös suorittaa ohjelmassa haarautumisia Z-lipun lailla. BCS suorittaa haarautumisen, jos C-lippu on asettunut ja BCC haarauttaa vain, jos C-lippu on nolla.

Kaikki käskyt eivät vaikuta lippujen sisältöön. On käskyjä, jotka jättävät ne rauhaan ja siten lippujen tilaa voidaan tutkia myöhemminkin kuin välittömästi seuraavassa käskyssä. On kuitenkin oltava tarkkana, ettei päästä lippuja lojumaan liian kauan ennen niiden tilaa testaavaa käskyä.

Kirjan lopussa olevassa assembly-kielen käskyjä kuvaavassa taulukossa on aina mainittu, mihin lippuihin käskyt vaikuttavat. Jos olet epävarma siitä, vaikuttaako viimeisin käsky lippuihin varmasti, on olemassa käsky, jolla voit varmistaa vaikutuksen: CMP.

CMP on vertailukäsky, joka vähentää akun sisällöstä joko välittömästi jonkin luvun (# luvun edessä) tai jostain muistipaikasta (\$) ja heksaosoite). Käsky ei kuitenkaan vaikuta akun sisältöön millään lailla: vähennys suoritetaan, mutta tulos jää vain ilmaan roikkumaan ilman, että sitä siirrettäisiin akun uudeksi arvoksi.

CMP-käskystä löytyy selitys myös sille, miksi esimerkiksi nollalippua testaava haarautumiskäsky on nimeltään BEQ (Branch if Equal = haaraudu, jos yhtä suuri). Akku saa sisältää minkä tahansa luvun ja jos sitä verrataan CMP-käskyllä yhtä suureen lukuun, on koneen sisällä suoritettavan vähennyslaskun tulos nolla ja nollalippu asettuu, vaikka akku ei ollutkaan nolla, vaan YHTÄ SUURI kuin verrattava luku. Haarautumiskäskyistä ja akulla suoritettavista laskuoperaatioista opit lisää seuraavissa luvuissa.

## Ripheet

Tässä olivat kaikki keskusyksikön tärkeät rekisterit. Kaikkia niitä et voi konekielelläkään ohjelmalla käsitellä, mutta kaikki liittyvät ohjelman toimintaan oleellisella tavalla.

Mikroprosessorin sisällä on kuitenkin vielä toimintoja, jotka aiheuttavat vastaamatta jääviä kysymyksiä, jos emme tutustu lyhyesti myös niihin. Olet varmaan ihmetellyt, kuinka yksittäiset muistipaikat tietävät oman osoitteensa tunnistaessaan, haluaako keskusyksikkö TALLENTAA niihin luvun vai haluaako se LUKEA niiden sisältämän luvun.

Tämän tiedon keskusyksikkö välittää muistipaikoille yhden sähköjohtimen

kautta. Tätä johdinta sanotaan R/W-linjaksi (luku/kirjoituslinja). Kun tämä johdin on nollatilassa (jännitteetön) muisti tietää, että keskusyksikkö haluaa tal-  
lenta luvun muistiin eli kirjoittaa. Kun linja on ykköstilassa (jännitteellinen) on  
vuorossa muistista lukeminen.

Muistien toimintaan liittyy muitakin, lähinnä ajoituksen kanssa tekemisissä  
olevia signaaleja, mutta ne eivät toiminnan tuntemisen kannalta ole niinkään  
tärkeitä.

Keskusyksikön yhdistää ulkomaailmaan vielä muutama johdin, joilla on mer-  
kitystä edistyneemmille ohjelmoijille: NMI ja IRQ. Ne ovat keskeytyspyyntölin-  
joja. Keskeytykset tarkoittavat sitä, että havaittuaan esimerkiksi jonkin oheislait-  
teen pyytävän keskeytystä jompaa kumpaa johdinta pitkin, keskusyksikkö kes-  
keyttää sen hetkisen ohjelman suorittamisen. Se pistetään ikään kuin pöydälle  
odottamaan, ja välissä suoritetaan tietyistä osoitteista alkavat keskeytysohjelman  
käskyt. Ne suoritettuaan keskusyksikkö jatkaa taas alkuperäistä työtään varsinais-  
en ohjelmansa parissa.

Tietokoneesi suorittaa näitä keskeytyksiä itse asiassa kaiken aikaa. Muun mu-  
assa koneen sisäinen kello pidetään niiden avulla päivitettyinä. Niiden toimintaa  
ei kuitenkaan huomaa, eivätkä omat konekieliset ohjelmasikaan pahastu näistä  
keskeytyksistä. Ne tulevat ja menevät ja kestävät yleensä niin vähän aikaa, ettei-  
vät ne vaikuta edes ohjelman suoritusnopeuteen.

Edistynyt ohjelmoija voi kuitenkin käyttää näitä keskeytysominaisuuksia hy-  
väkseen. Ne mahdollistavat esimerkiksi kahden ohjelman samanaikaisen ajami-  
sen. Keskeytysohjelmaksi voi tehdä musiikkia tuottavan konekielisen rutiinin,  
joka toimii taustalla, vaikka ajaisit samalla omaa konekieliohjelmiasi tai vaik-  
kapa Basic-ohjelmaa. Basic-ohjelmakaan ei häiriinny taustalla pyörivästä mu-  
siikkiohjelmasta lainkaan vaan toimii normaalisti edeten käsky käskyltä eteen-  
päin.

Eräs keskusyksikön alue on vielä hyvä tuntee. Se on aritmeettis-looginen yk-  
sikkö eli tuttavallisesti ALU. Se suorittaa kaikki laskentaan vivahtavat tehtävät  
keskusyksikön sisällä: yhteenlaskut, inkrementoinnit, dekrementoinnit ja loogiset  
operaatiot, joista enemmän tuonnempana tässä kirjassa.

Mikään rekisteri ei sellaisenaan pysty suorittamaan laskentaa vaan kaikki luvut  
käyvät ensin ALUssa manipuloitavina ja tulokset sijoitetaan takaisin kyseisiin  
rekistereihin.

ALU ei matemaattisilta ominaisuuksiltaan ole mikään nero, kuten ei koko  
keskusyksikkökään. Se pystyy suorittamaan 8 bitin yhteenlaskuja ja vähennyslas-  
kuja, siirtämään luvun bittejä yhden bitin verran oikealle tai vasemmalle ja suo-  
rittamaan loogisia operaatioita kuten AND (JA), OR (TAI) ja EOR (ehdoton  
TAI). Nämä operaatiot ja niitä vastaavat käskyt muuttavat akkurekisterin sisäl-  
tämän luvun yksittäisiä bittejä halutulla tavalla.

Ohjelmoija saa kuitenkin näilläkin vähillä laskennallisilla ominaisuuksilla

paljon aikaan. Kertolaskuhan on vain toistettua yhteenlaskua ja vastaavasti jakolasku on toistettua vähennyslaskua. Tietokone saadaan ratkaisemaan vaikka kuinka monimutkaisia matemaattisia kaavoja, kunhan sen pienet käskyt vain si-  
joitetaan oikeaan järjestykseen. Avaruusrakettien ratojen laskeminen on eräs monimutkaisimmista nykyajan matemaattisista toiminnoista ja ne on mahdollista tehdä vain tietokoneilla, ihminen olisi siihen liian epäluotettava ja hidas.



# 5



## Posti kulkee

Merkittävin käytännön eroavuus Basicin ja konekielen välillä on osoitetietojen käsittely. Basic hoitaa suuren osan rutiininomaisesta asioiden sijoittamisesta ja muistin alueiden varaamisesta. Basicisti viettää itse asiassa varsin leppoisia päiviä tehdessään ohjelmaansa uusia rivejä, lisätessään siihen muuttujia.

Basicissa kaikella on symboliset nimensä. Ohjelmariveillä numerot, muuttujilla usein selväkieliset nimet. Basic-ohjelmoijan ei tarvitse kantaa huolta siitä, missä kukin ohjelmarivi muistissa sijaitsee ja missä pidetään mitään muuttujaa. Basic-tulkki huolehtii näistä tehtävistä automaattisesti. Ohjelmaan voi lisätä väliin rivejä ja muuttujiin voi viitata pelkästään käyttämällä niiden nimiä.

Konekielessä tilanne on täysin erilainen. Jotakin on maksettava konekielen tuomasta nopeudesta ja monipuolisista mahdollisuuksista. Merkittävin uhraus on se, että ohjelmoijan on itse pidettävä yllä tietoutta siitä, missä hänen muuttujansa sijaitsevat. Jokainen muistipaikka on periaatteessa mahdollinen muuttuja.

Konekielessäkin on käskyllä eräänlainen rivinumero. Sitä vastaa käskyn todellinen osoite muistissa. Kahden käskyn väliin ei automaattisesti voi lisätä yhtä käskyä, vaan se vaatii koko jäljestä tulevan käskyjonon siirtämistä tilan synnyttämiseksi.

Edistyneemmät konekielen apuneuvot, assembler-kääntäjät, tosin paikkaavat suurelta osin tätä konekielen kyvyttömyyttä käsitellä muuttujia symboleina eli niminä. Assemblerit sallivat käyttäjänsä yleensä antaa muistipaikoille ja ohjelmariveille nimiä ja näihin paikkoihin voi tämän jälkeen viitata nimen avulla. Basicissakaan ei ole tällaista ominaisuutta jotain yksittäistä poikkeusta lukuunottamatta. Basicissa on hyppykäskyihin aina annettava rivinumero, joka toimii Basicissa ohjelmarivin nimenä.

Yhtä kaikki, assemblerinkin tuomista hienoista ohjelmointiominaisuuksista

huolimatta konekielessä on kuitenkin osattava päättää, minne ohjelmansa muistissa sijoittaa. Ohjelmoijan on myös varattava tietty riittävä alue muuttujilleen jo ennen varsinaista ohjelman kirjoittamista.

## Osoitusmuodot

Vaikka assembleria käytettäessä voitaisiin viitata osoitteisiin nimen perusteella on jokaisen käskyn yhteydessä kuitenkin osattava valita myös käskyn osoitusmuoto. Jokainen konekielen käsky kohdistuu johonkin paikkaan tietokoneen sisällä, joko muistiin tai keskusyksikön sisäiseen rekisteriin. Koneelle on osattava selvittää, mitä niistä tarkoitetaan.

Kullakin käskyllä on erilaisia tapoja, millä se voi viitata kohdepaikkaansa. Nämä tavat ovat osoitusmuotoja, joita 6502:ssa on yhteensä 13 erilaista. Valtaosa niistä keskittyy muistipaikkojen osoitteiden selvittämiseen, keskusyksikön sisällä rekistereillähän ei varsinaisia osoitteita olekaan.

Muutamaan osoitusmuotoon olet jo aiemmissa luvuissa tutustunutkin. Enimmäkseen ovat esillä olleet suoraviivaisimmat muodot eli välitön osoitusmuoto ja absoluuttinen osoitusmuoto.

## Välitön (Immediate)

Välittömän osoitusmuodon perusesimerkki on käsky LDA #\$05. Se lataa akkuun välittömästi käskyä LDA seuraavan luvun eli heksaluvun 05. Risuaitamerkki (#) luvun edessä kertoo, että kyseessä on juuri välitön osoitusmuoto.

Välitöntä osoitusmuotoa käytettäessä siirretään jokin tunnettu vakio joko akun tai jonkin muun keskusyksikön sisäisen rekisterin sisällöksi (aiempi sisältö katoaa samalla). Koska ohjelmoijan täytettävissä olevat rekisterit ovat kaikkia 8-bittisiä, voidaan latauskäskyllä siirtää ainoastaan lukuja 0 — 255 eli heksana ainoastaan kaksinumeroisia lukuja (\$00 — \$FF). Välitön osoitus vastaa LDA:n yhteydessä aivan Basicin sijoituskäskyä, esimerkiksi

**10 LET K=5**

jossa muuttujan K arvoksi sijoitetaan välittömästi seuraava luku eli 5.

Välitöntä osoitusmuotoa voi käyttää latauskäskyjen lisäksi myös lisättäessä, vähennettäessä ja vertailtaessa akun sisältöä johonkin lukuun.

Ole tarkkana risuaitamerkin kanssa tätä osoitusmuotoa käytettäessä. Se saattaa joskus helposti unohtua ja käskyn perään tulee laitettua vain \$05. Silloin assembler-ohjelmasi luulee sinun tarkoittavan osoitetta \$05 ja se valitsee ohjelmaan sijoitettavaan käskyyn toisen osoitusmuodon.

Kun teet lyhyitä ohjelmia ja koodaat konekieltä käsin, ei tällaista virhettä pääse tapahtumaan vaan valitset taulukosta suoraan oikean käskykoodin. Itse muistiinhan ei risuaitamerkkiä laiteta (eikä myöskään dollarimerkkiä). Nämä erikoismerkit ovat mukana ainoastaan kertomassa assembler-ohjelmalle, mitä osoi-



tusmuotoa tarkoitamme ja missä järjestelmässä antamamme luku on. Assembler-ohjelma valitsee sitten antamiemme merkkien mukaan oikean käskykoodin ja muuttaa antamamme luvun keskusyksikön ymmärtämään muotoon (kaksijärjestelmän ykkösiksi ja nolliksi).

## Absoluuttinen (Absolute)

Absoluuttinen osoitusmuoto on varsin selväpiirteinen. Siinä annetaan käskyn perään suoraan ja yksiselitteisesti sen muistipaikan osoite, jota käskyssä tarkoitetaan. Esimerkiksi

**LDA \$4F00** (lataa akkuun luku, joka sijaitsee muistipaikassa \$4F00)

tai

**DEC \$4F00** (vähennä yksi luvusta, joka löytyy osoitteesta \$4F00. Luku säilyy yhdellä vähennettynä tässä osoitteessa)

Absoluuttiselle osoitusmuodolle löytyy myös hyvä vastine Basicista:

**10 LET K = M**

Tässä Basic-käskyssä sijoitettiin K:n arvoksi muuttujan M arvo aivan samoin kuin LDA \$2000 sijoittaa akkuun muuttujan eli muistipaikan \$2000 sisällön.

Absoluuttisen osoituksen argumentti eli käskyä seuraava luku on aina neljän heksaluvun mittainen eli kaksi tavua. Tämä sen vuoksi, että sillä pystyttäisiin osoittamaan erikseen jokaista muistipaikkaa koneen sisällä.

## Nollasivu (Zero Page)

Absoluuttista osoittamista hyvin lähellä on nollasivun osoittaminen. Se ilmaistaan assemblerille antamalla osoitteeksi vain kaksinumeroinen heksaluku, esimerkiksi LDA \$05.

Tietokoneen muisti on yhteensä 65536 tavua pitkä. Sen ajatellaan koostuvan 256 tavun mittaisista lohkoista, joita kutsutaan sivuiksi. Nollasivu on näistä ensimmäinen ja se sijaitsee osoitteissa \$00 — \$FF (0 - 255). Kahden heksanumeron avulla voimme juuri kattaa tämän muistialueen.

Minkä takia on olemassa tällainen nollasivun osoitusmuoto? Emmekö voisi aina yhtä hyvin käyttää selvyiden vuoksi kokonaista absoluuttista osoitetta? Voimme toki käyttää aina absoluuttistakin osoittamista, myös nollasivulle viitataessa. Jos kirjoitat assemblerillesi LDA \$0005, saattaa se valita absoluuttisen osoitustavan ja sijoittaa osoitteen ohjelmaan kahden tavun mittaisena (jokin assembler-ohjelma tosin saattaa älykkäästi huomata, että hopsan! tekipäs kaveri tyhmästi kun ei käyttänyt nollasivua, ja sijoittaa käskyn ohjelmaan nollasivun osoituksessa poistaen osoitteesta turhat etunollat).

Syy siihen, miksi tällainen lyhennetty osoitusmuoto on olemassa on juuri sen lyhyys ja sen tuoma nopeus. Aikanaan, kun tietokoneen muisti oli kallista ja ny-

kyisten kotitietokoneiden 64 kilotavun muistista vasta uneksittiin suurissa tietokoneissa, käytettiin mieluusti tällaista muistia säästävää käskymuotoa.

Nykyään muistia on tietokoneissa aivan riittävästi eikä muistitila ole enää mikään peruste yhden tavun säästämiseksi. Nopeus saattaa sen sijaan joskus olla hyväkin peruste käyttää nollasivun osoitusta. Itse asiassa tietokoneesi käyttöjärjestelmä käyttää ahkerasti nollasivun muistipaikkoja hyväkseen.

Kun teet ohjelmassasi silmukan (toistuvasti suoritettava ohjelmanpätkä), jota kierretään todella monta kertaa, sijoita siinä tarvittavat muuttujat nollasivulle. Käyttöjärjestelmä tosin varaa nollasivusta itselleen leijonan osan samaten kuin Basic-tulkki, joten ole varovainen. Selvitä oman koneesi tiedoista, mistä päin nollasivua löytyy vapaata tai turvallista säilytystilaa omille muuttujillesi.

Nollasivun osoituksessa piilee eräs konekieliohjelmoinnin kompastuskivistä. Usein ohjelmoijalta unohtuu välittömän osoitustavan vaatima risuaitamerkki (#) pois ja käskystä tuleekin tyyppiä LDA \$05. Assembler-ohjelma tulkitsee sen autuaan onnellisena nollasivun osoitukseksi ja sijoittaa muistiin käskyn, joka haakee luvun OSOITTEESTA \$05 eikä sijoita suoraan LUKUA \$05, mikä kuitenkin oli tarkoituksena.

Nollasivu on siis erikoisasemassa tietokoneen muistialueella ja nollasivun osoitus on erikoistapaus absoluuttisesta osoittamisesta.

## Absoluuttinen,X ja Absoluuttinen,Y (Absolute,X Absolute,Y)

Tämä osoitustapa on erittäin käyttökelpoinen selattaessa listoja ja taulukoita. Se on muunnelma absoluuttisesta osoituksesta, mutta tässä tapauksessa osoitteeseen on liitetty X- tai Y-rekisterin arvo. Käsky

### LDA \$2000,X

ei haakaan akkuun lukua osoitteesta \$2000 vaan osoitteesta, joka saadaan, kun \$2000:een LISÄTÄÄN X-rekisterin sen hetkinen arvo. Vastaavasti voidaan toimia myös Y-rekisterin kanssa. Jos edellisessä käskyesimerkissä X-rekisterin arvo olisi 5, haettaisiin akkuun luku osoitteesta \$2005.

Tämän käskyn voima perustuu siihen, että X-rekisterin arvoa voidaan käskyn suorittamisen jälkeen muuttaa, esimerkiksi kasvattaa yhdellä. Sama käsky voidaan ohjelmassa haarautumalla suorittaa toistamiseen ja useamminkin aina eri X-rekisterin arvoilla. Liu'uttamalla X (tai Y) -rekisterin sisältämää lukua ylöspäin tai alaspäin voimme helposti muutamana käskyn puitteissa liikkua läpi suurenkin taulukon ilman, että latauskäskyä tarvitsisi kirjoittaa uudestaan jokaiselle muistipaikalle.

Esimerkkinä suunnittelemme konekielisen ohjelman, joka siirtää muistista kuvaruutumuistin alueelle kokonaisen kuvaruudullisen tekstiä. Samalla tämä ohjelma antaa hyvän demonstraation konekielen nopeudesta. Tehtävä sujuu sil-

manräpäyksessä, mihin Basic ei suinkaan pysty. Kuvaruudun alkuosoite (\$0400) vastaa Commodore 64:n arvoa.

- \$0350 LDX #\$00** (lataa X-rekisteriin luku 0)
- \$0352 LDA \$D000,X** (lataa akkuun merkki osoitteesta \$D000 + X. Ensimmäisellä kerralla X on nolla, seuraavalla 1 jne.)
- \$0355 STA \$0400,X** (sijoitamme merkin kuvaruutumuistiin. Ensimmäisellä kerralla merkki menee kuvaruudun ensimmäiselle paikalle, toisella kerralla toiselle paikalle jne. Huomaa, että X-rekisterillä on tässä kaksoistehtävä: se juoksee haettavien merkkien mukaan arvosta toiseen, mutta tässä käskyssä se juoksee pitkin kuvaruutumuistin paikkoja, jonne merkit jälleen sijoitetaan)
- \$0358 LDA \$D100,X** (koska kuvaruutumuisti on 1000 merkkiä pitkä ja X-rekisterin 8-bittisellä sisällöllä voimme liukua vain 256 tavun yli, kirjoitamme tämän ohjelmanpätkän 4 kertaa peräkkäin, joista kukin hoitaa oman 256 tavun eli yhden muistisivun siirtämisen. Tämä siirrettävän blokin osoite poikkeaa \$0100:lla edellisestä, joka on juuri 256 10-järjestelmässä)
- \$035B STA \$0500,X** (ensimmäisellä kierroksella tämä käsky sijoittaa merkin jonnekin kuvaruudun seitsemännellä rivillä. Saman ohjelmanpätkän seuraavat toisinnot aloittavat täyttämisen vastaavasti 13. riviltä ja 20. riviltä. Kuvaruutua täytetään siis yhtäaikaa neljästä kohdasta)
- \$035E LDA \$D200,X** (tämä osa siirtää kolmannen sivun kuvaruudulle. Kun ajat ohjelman havaitset, ettei kuvaruudulle ilmesty mitään järkevää. Se ei ole tarkoituskaan: siirrämme kuvaruudulle ROM-muistin sisältöä. Jos meillä olisi jossain muistilohkossa sijoitettuna järkevä kuvaruudullinen tekstiä tai vaikkapa merkkigrafiikkaa, sujuisi sen siirtäminen tällä ohjelmalla vaivattomasti)
- \$0361 STA \$0600,X** (tämä käsky täyttää kuvaruutua 13. riviltä alkaen)
- \$0364 LDA \$D300,X** (tämä hakee viimeisen 256 tavun lohkon.)
- \$0367 STA \$0700,X** (tämä käsky alkaa täyttää kuvaruutua 20. riviltä lähtien, 8. sarakkeesta. Neljällä 256 tavun siirrolla tulee siirrettyä yhteensä 1024 tavua paikasta toiseen. Kuvaruutu on vain 1000 tavun kokoinen (25riviäx40merkkiä), joten 24 tavua on ylimääräistä. Niiden siirtäminen jonon jatkona ei kuitenkaan häiritse ketään,ellemme käytä kuvaruutumuistin välitöntä jatkoa johonkin muuhun)
- \$036A INX** (nyt kasvatetaan X:n arvoa yhdellä ja siirrytään kussakin neljässä siirtolohkossa pykälällä eteenpäin)

- \$036B BNE \$0352** (jos edellisessä käskyssä X-rekisteriä kasvatettiin 256. keran, se kasvoi 'yli laitojensa'. Kun 8-bittisen rekisterin sisältöön \$FF lisätään vielä yksi, se muuttuu jälleen nollassi. BNE käskyllä testataan, muuttuiko X-rekisteri nollassi. Jos ei vielä, palataan osoitteeseen \$0352 ja jatketaan siirtoja. Jos X-rekisteri tuli jo nollassi jälleen ei enää palata vaan siirrytään normaalisti seuraavaan käskyyn)
- \$036D RTS** (homma on hoidettu, palataan takaisin sinne mistä lähdettiin)

Edellinen esimerkki vastaa 25 Basic-ohjelmariiviä, joissa kussakin on 40 merkkiä tulostava PRINT-lause. Kummallakin saadaan kuvaruutu täyteen tavaraa. Basicissa 'tavara' sisältyy PRINT-lauseisiin, mutta konekielessä meidän on muulla tavalla sijoitettava se ensin muistiin jollekin vapaalle alueelle.

### Nollasivu,X (Zero Page,X)

Tämä osoitusmuoto noudattaa samaa logiikkaa absoluuttinen,X:n suhteen kuin vastaavat osoitusmuodot ilman X-lisuketta (absoluuttinen ja nollasivu). STA \$80,X tallettaa akun arvon nollasivulle osoitteella, johon on lisätty X-rekisterin arvo.

Jälleen assemblerimme tunnistaa antamastamme lyhyestä, kahden tavun osoitteesta, että tarkoitamme nollasivun osoitusta ja se valitsee muistiin sijoitettavaksi sitä vastaavan STA-käskykoodin. STA-käskyllä on seitsemän erilaista koodia, jotka keskusyksikkö ymmärtää saman käskyn eri osoitusmuodoiksi. Assembler-ohjelmamme valitsee näistä sen oikean sen perusteella, miten me argumentin eli käskyn jälkeisen osoitustavan merkitsemme. On sentään huomattavasti helpompaa muistaa vain yksi käsky, STA, kuin seitsemää eri numerosarjaa. Tämä on assemblerohjelman käytön tuoma voimakas hyöty.

### Nollasivu,Y (Zero Page,Y)

Y-rekisterin käyttö indeksinä nollasivun osoituksen yhteydessä vastaa täysin nollasivu,X:n käyttöä. Sitä voidaan kuitenkin käyttää vain LDX- ja STX-käskyjen yhteydessä (LDX = lataa X-rekisteriin, STX = tallenna X-rekisterin sisältö).

Syy tällaiseen rajoitukseen löytyy keskusyksikön sisältä: se ei vain osaa käyttää tätä osoitusmuotoa muiden käskyjen yhteydessä. Jostain syystä niitä toimintoja ei ole keskusyksikköön rakennettu. Ehkäpä mikropiirin kapasiteetti on ollut tulossa täyteen tai sen turhaa monimutkaisuutta on haluttu muusta syystä välttää. Mene ja tiedä. Meidän tarvitsee vain muistaa, että nollasivu,Y:tä ei voi käyttää muualla kuin LDX- ja STX-käskyjen yhteydessä.

## Epäsuora, Y (Indirect, Y)

Epäsuorat osoitustavat monimutkaistavat ajatusrakennelmaamme osoittamisesta entisestään: välittömässä osoituksessa sijoitettava luku tuli heti itse käskyn perässä, ohjelman sisällä. Absoluuttisessa osoituksessa tiesimme osoitteen, josta lukua saatoimme hakea. Epäsuorassa osoituksessa emme tiedä edes osoitetta vaan sekin on haettava ensin muistista.

Y- ja X-rekisterin käyttö epäsuorassa osoituksessa on erilaista. Pidä mielessäsi, että ne ovat kaksi aivan eri muotoa.

LDA (\$05), Y tarkoittaa seuraavaa: haemme muistipaikasta \$05 osoitteen, lisäämme siihen Y-rekisterin arvon ja käyttämällä tätä osoitteena haemme luvun, jonka lataamme akkuun. Monimutkaista, eikö totta? Älä hätäänny, jos et saa selvennettyä itsellesi täysin sen toimintaa vielä tässä vaiheessa. Esimerkkien valossa tapahtuva kertaaminen myöhemmin saa sinut vielä tähänkin "sisälle".

Eräs asia jäi ehkä kaivamaan mieltäsi: osoitehan on yleensä kaksi tavua pitkä. Kuinka se voidaan hakea yhden tavun muistipaikasta osoitteesta \$05? Siinäpä se. Äsken ei tullut kerrotuksi aivan kaikkea kerralla. Osoite haetaan itse asiassa kahdesta tavusta: muistipaikasta \$05 ja \$06, ja vieläpä siten, että \$05:ssä sijaitsee hakemamme osoitteen alempi tavu eli vähemmän merkitsevä osuus ja muistipaikassa \$06 sijaitsee sen enemmän merkitsevä tavu. Keskusyksikkömme 6502 kun on sen verran omituinen, että se haluaa saada osoitteet tällä tavalla käänteisessä järjestyksessä.

Koska haetun osoitearvon toinen tavu sijaitsee aina seuraavassa muistipaikassa, ei käskyn yhteydessä tarvitse kertoa kuin ensimmäisen tavun sijainti. Keskusyksikkö osaa automaattisesti hakea toisen tavun seuraavasta osoitteesta. Eräs rajoitus tälläkin käskyllä on: perusosoite, josta varsinainen osoite haetaan, voi sijaita ainoastaan nollasivun alueella eli osoitteissa 0 — 255 tai heksana \$00 — \$FF

Sulut ilmaisevat tässä käskyssä assembler-ohjelmallemme, että käskykoodien joukosta on valittava se, joka aiheuttaa epäsuora, Y -osoituksen. Huomaa myös, että sulut merkitään vain perusosoitteen ympärille ja että perusosoite on vain kaksi heksanumeroa pitkä (vain nollasivu).

Tämä osoitusmuoto antaa varsin laajat muistinkäsittelymahdollisuudet.

Liu'uttamalla Y-rekisterin arvoa 0:n ja 255:n välillä voimme juosta yhden muistisivun sisällä nollasivulle tallennetun osoitteen viitoittamassa paikassa. Lisäämällä esimerkkikäskyssä muistipaikkaan \$06 (se enemmän merkitsevä tavu) yhden, voimme siirtyä juoksemaan seuraavan muistisivun alueelle jne.

Edellinen esimerkkiohjelma, joka siirsi ruudun täyteen tekstiä, olisi voitu kirjoittaa toisella tavalla tätä osoitusmuotoa hyväksikäyttäen. Silloin meidän ei olisi tarvinnut kirjoittaa neljää kertaa peräkkäin lataus- ja tallennuskäskyjä, vaan olimme voineet siirtyä sivun verran eteenpäin pitämällä lähtöalueen osoitetta ja kuvaruutumuistin alkuosoitetta kumpaakin kahdessa nollasivun tavussa.

Esimerkkiohjelmamme näyttäisi silloin tällaiselta. Koeta lukea ohjelmaa läpi,

ja yritä selvittää itsellesi, miten se toimii ja hyppää yhden sivun täytettyään toiselle sivulle.

|        |              |                                                                                                                   |
|--------|--------------|-------------------------------------------------------------------------------------------------------------------|
| \$0350 | LDA #\$D0    | (ladataan akkuun lähtöalueen osoitteen enemmän merkitsevä tavu)                                                   |
| \$0352 | STA \$FC     | (viedään se osoitteeseen \$FC nollasivulla)                                                                       |
| \$0354 | LDA #\$00    | (ladataan akkuun lähtöalueen osoitteen vähemmän merkitsevä tavu)                                                  |
| \$0356 | STA \$FB     | (viedään se osoitteeseen \$FB nollasivulla)                                                                       |
| \$0358 | LDA #\$04    | (ladataan akkuun kuvaruutumuistin alkuosoitteen merkitsevämpi tavu)                                               |
| \$035A | STA \$FE     | (viedään se muistipaikkaan \$FE nollasivulla)                                                                     |
| \$035C | LDA #\$00    | (ladataan akkuun kuvaruutumuistin alkuosoitteen vähemmän merkitsevä tavu)                                         |
| \$035E | STA \$FD     | (viedään se muistipaikkaan \$FD nollasivulla)                                                                     |
| \$0360 | LDX #\$04    | (lataa X-rekisteriin siirrettävien sivujen lukumäärän)                                                            |
| \$0362 | LDY #\$00    | (lataa nollan Y-rekisteriin, joka toimii juoksupoikana liukuen 0:sta 255:een)                                     |
| \$0364 | LDA (\$FB),Y | (haetaan lähtöalueelta luku)                                                                                      |
| \$0366 | STA (\$FD),Y | (viedään se kuvaruutumuistiin)                                                                                    |
| \$0368 | INY          | (kasvatetaan Y-rekisterin arvoa yhdellä)                                                                          |
| \$0369 | BNE \$0364   | (jos Y ei vielä tullut uudestaan nollaksi, haaraudutaan takaisin siirtämään seuraava tavu)                        |
| \$036B | INC \$FC     | (yksi sivu tuli siirrettyä, siirrytään seu raavalle sivulle kasvattamalla osoitteen                               |
| \$036D | INC \$FE     | eniten merkitseviä tavuja yhdellä)                                                                                |
| \$036F | DEX          | (vähennetään sivulaskuria yhdellä)                                                                                |
| \$0370 | BNE \$0362   | (jos siirrettäviä sivuja on vielä jäljellä, siirrytään osoitteeseen \$0362 ja aloitetaan uuden sivun siirtäminen) |
| \$0372 | RTS          | (palataan sinne, mistä tähän ohjelmaan tulimme)                                                                   |

Huomaat varmaan, että tällä tavalla saimme aikaan pidemmän ohjelman kuin aiemmin. Aina ei siis kannata käyttää "sofistikoituja" osoitusmuotoja. Monimutkainen osoitusmuoto vaatii myös monimutkaista tehtävää, ja sitä ei siirtoesimerkkimme ollut.

## Epäsuora,X (Indirect,X)

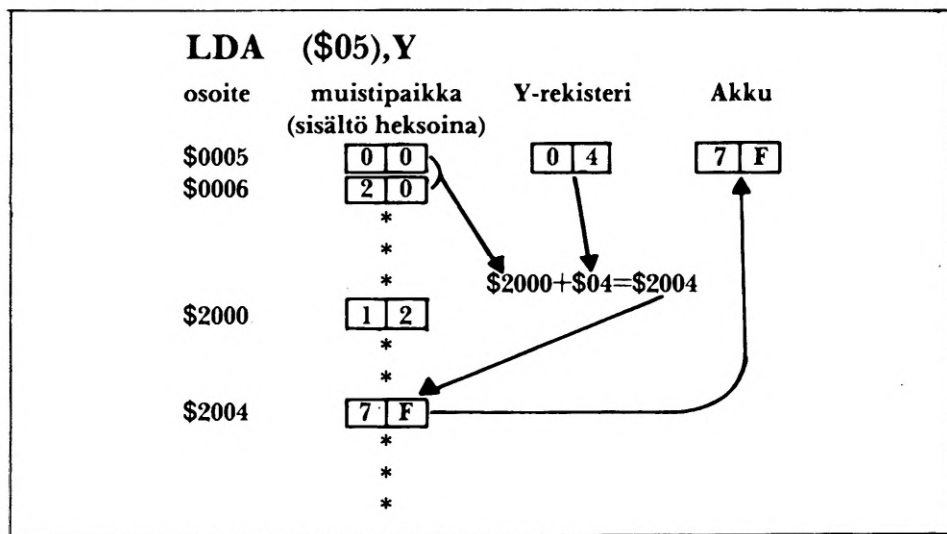
X-rekisterin yhteydessä epäsuora osoitusmuoto toimii eri tavalla. LDA (\$05,X) EI lisää muistipaikoista \$05 ja \$06 löytyvään osoitteeseen mitään. Tämä käsky lisää X:n arvon itse osoitteeseen \$05! Eli, jos käskyä LDA (\$05,X) suoritettaessa X-rekisteri sisältää luvun 2, lisätään se osoitteeseen \$05, jolloin osoitteeksi tulee \$07. Kun kyseessä oli EPÄSUORA osoitusmuoto, ei tämä osoite ole lopullinen,

vaan lopullinen osoite löytyy sen osoittamasta muistipaikasta, tarkemmin sanoen muistipaikoista \$07 ja \$08. 6502:n tavan mukaan vähemmän merkitsevä osoitetavu on \$07:ssä ja enemmän merkitsevä \$08:ssa.

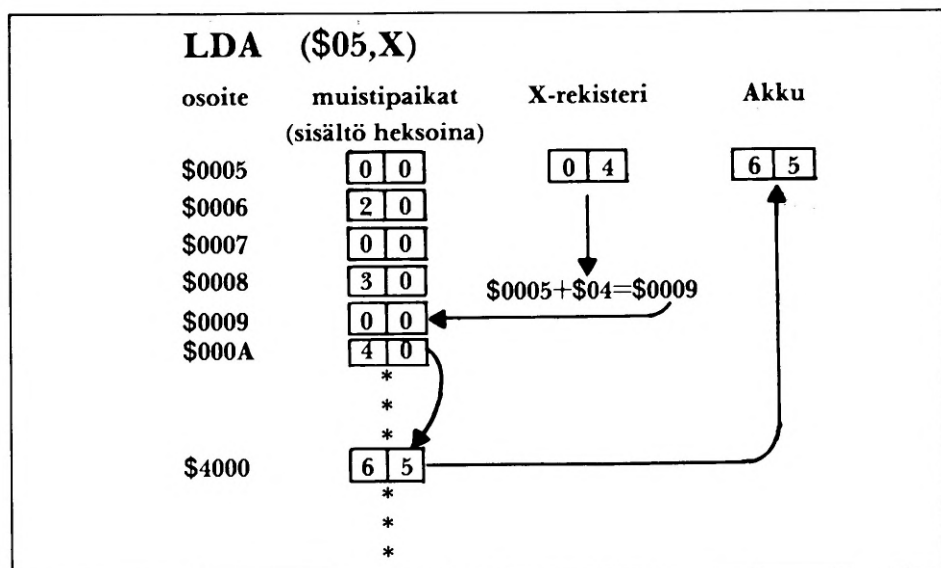
Huomaa käskyn merkintätavassa, että X on sulkujen sisällä. Epäsuora, Y -osoitusmuodossa Y ei ollut sulkujen sisällä. Tämä merkintätapa auttaa muistamaan, että X:n yhteydessä X:n arvo lisätään jo ensimmäiseen osoitteeseen (joka on käskyssä mukana) ja Y:n yhteydessä Y:n arvo lisätään vasta siihen osoitteeseen, joka tästä muistipaikasta löytyy.

Epäsuora, X -osoitusmuodon merkitys on melko pieni. Sen avulla saadaan rakennetuksi osoitteistotaulukoita, jolloin X:n arvosta riippuen käskyn vaikutus voidaan kohdistaa eri puolille keskusyksikön muistialuetta. Edellä useampaan kertaan esillä ollut esimerkki tekstilohkon siirtämisestä kuvaruutumuistin alueelle voitaisiin toteuttaa tälläkin osoitustavalla, mutta sekin olisi ensin esittämäämme yksinkertaista toteutusta pitempi, joten ei ole mieltä rakentaa enää toista huonompaa vaihtoehtoa.

Katsokaamme sen sijaan tarkemmin, miten epäsuorat käskymuodot toimivat:







Osoitusmuodossa epäsuora,Y lisätään Y-rekisterin sisältö lopulliseen osoitteeseen. Osoitusmuodossa epäsuora,X lisätään X-rekisteri siihen osoitteeseen, josta lopullista osoitetta haetaan.

On olemassa myös epäsuora osoitusmuoto ilman mitään indeksirekisteriä liitettyinä osoitteen yhteyteen. Sitä voidaan käyttää ainoastaan ehdottoman hyppykäskyn, eli JMP:n, yhteydessä (JMP: JuMP = hyppää). JMP vastaa Basicin GOTO-käskyä. Sillä voi hypätä joko suoraan absoluuttiseen osoitteeseen (esimerkiksi JMP \$2000, jolloin ohjelma siirtyy suorittamaan ohjelmaa osoitteeseen \$2000) tai epäsuorana (esimerkiksi JMP (\$2000), jolloin ohjelma siirtyy suorittamaan käskyä siitä osoitteesta lähtien, joka löytyy käskyssä annetusta osoitteesta \$2000 ja \$2001, vähemmän merkitsevä osoitteenosa ensimmäisessä tavussa).

Epäsuoraan osoitusmuotoon liittyy kuitenkin mikroprosessorin sisäinen toimintavirhe, jonka takia sen käyttöä on tarkoin harkittava. Jos nimittäin sulkujen sisällä antamasi osoite osuu muistisivun yläreunaan, eli on muotoa \$??FF, jossa kysymysmerkkien paikalla voi olla mitä tahansa heksalukuja, PITÄISI keskusyksikön hakea lopullisen osoitteen ensimmäinen tavu osoitteesta \$??FF ja toinen eli merkitsevämpi tavu yhtä suuremmasta osoitteesta.

Näin ei kuitenkaan käy. Keskusyksikkö osaa aivan hienosti hakea lopullisen osoitteen ensimmäisen osan annetusta osoitteesta, mutta kun se muodostaa toisen osan osoitetta, se laskee pieleen. FF on alkuperäisen osoitteen vähemmän merkitsevä tavu ja kun siihen lisätään yksi pitäisi sen tavun pyörähtää jälleen lukualu-



ensa alkuun ja enemmän merkitsevään tavuun pitäisi lisätä 1. Sitä mikroprosessorin sisäinen elektroniikka ei kuitenkaan tee ja lopullisen osoitteen merkitsevämpi tavu haetaan aivan väärästä paikasta. Jos annamme käskyn JMP (\$15FF) haetaan hyppyosoitteen ensimmäinen tavu aivan oikein paikasta \$15FF, mutta toinen osa haetaan paikasta \$1500!

Tämä ei ole mikään suuri murhe ohjelmoinnissa. On helppoa muistaa tämän käskyn yhteydessä välttää sivun yläreunaa. Sen sijaan se on erittäin hyvä esimerkki siitä, kuinka vaikeata on saada aikaan ehdottoman oikealla tavalla toimivia tietokoneohjelmia. Mikroprosessorin sisäinen elektroniikka toimii sekin tietynlaisen ohjelman mukaisesti. Jokainen antamasi konekielinen käsky aiheuttaa mikroprosessorin sisällä varsin monimutkaisen sarjan toimintoja: lukuja siirrelään, osoitteita haetaan, ja välillä lasketaan. Vaikka mikroprosessorin sisäistä "mikro"ohjelmaa on suunniteltu monen miestyövuoden ajan, ei tämä pikkuinen bugi (ohjelmavirhe) ole tullut testeissä esiin.

Samalla tavalla sinunkin rakentamasi ohjelmat, olivat ne sitten Basic-kielisiä tai konekielisiä, sisältävät melko varmasti virheitä. Basicissa virheet on helppo löytää virheilmoitusten avulla ja Basicin suuren vuorovaikutteisuuden tukemana (ohjelmaa voi korjata mielin määrin nopeasti ja ajaa ohjelma heti uudelleen).

Konekielessä virheiden etsiminen on huomattavasti hankalampaa. Virheilmoituksia ei ole odotettavissa kuin ainoastaan assembler-ohjelmalta, kun se huomaa jotain aivan sopimatonta. Yleensä käy kuitenkin niin, että ohjelmaa ajettaessa se vain yhtäkkiä sekoaa, kone menee lukkoon, ja ainoa keino virhetilanteen korjaamiseksi on kytkeä virta hetkeksi pois päältä.

Muista siis ehdottomasti aina tallentaa ohjelmasi kasetille tai levyille ennen kuin ensimmäistäkään kertaa yrität ajaa sitä!

## Suhteellinen (Relative)

Hankalin osoitusmuoto tehtäessä konekielistä ohjelmaa käsikoodauksella on ehdottomasti suhteellinen osoitusmuoto. Se toimii ainoastaan ehdollisten hyppykäskyjen (BNE, BEQ jne.) yhteydessä ja on itse asiassa ainoa osoitusmuoto, jota niiden yhteydessä voi käyttää.

Näitä hyppykäskyjä käytetään ohjelmassa pääsääntöisesti varsin lyhyisiin haarautumisiin ohjelman sisällä. Ohjelmasil mukoiden muodostaminen on eräs esimerkki. Niissä yleensä haaraututaan taaksepäin muutaman käskyn verran, jos jokin ehto ei vielä ole tosi (akku tai jokin muu rekisteri ei vielä ole nolla tms).

Suhteellisessa osoitusmuodossa emme annakaan käskyn perässä mitään todellista osoitetta, johon haarautuminen suoritetaan vaan niiden tavujen määrän, joiden YLI hyppäämme eteen- tai taaksepäin. Tämä ylitettävien tavujen määrä annetaan käskyn perässä tulevassa yhdessä tavussa.

Yksi tavu voi osoittaa lukua 0 — 255, niinhän olemme oppineet. Kuinka sitten voimme tietää, milloin haaraudumme taaksepäin ja milloin eteenpäin? Ratkaisu

löytyy tavun sisältä. Tavu koostuu 8 bitistä, joiden lukualue on juuri 0 — 255, mutta haarautumiskäskyn yhteydessä esitämme yli hypättävien tavujen lukumäärän vain seitsemällä bitillä. Keskusyksikkö tulkitsee kahdeksannen, eniten merkitsevän bitin osoittavan hypyn suuntaa. Jos kahdeksas bitti on ykkönen, hyppäämme taaksepäin — jos se on nolla, hyppäämme eteenpäin.

Tällä tavalla voimme siirtyä 127 askelta taaksepäin tai 128 askelta eteenpäin siitä paikasta, jossa olemme. Selvää, eikö totta? Mutta hankaluus tulee vasta tämän jälkeen. Eteenpäin suuntautuvat hyppy voi laskea aivan normaalisti, kunhan vain muistaa, että haarautumiskäskyä toteutettaessa ohjelmalaskuri osoittaa jo seuraavaa käskyä. Eli seuraava käsky on laskuissamme 0, sitä seuraava 1 jne.

Taaksepäin suuntautuvien hyppyjen yhteydessä luvut eivät menekään aivan loogisesti. Katsotaan ongelmaa esimerkin valossa. Tällä tavalla hyppykäsky kirjoitetaan assembleria käytettäessä:

```
$2000 LDX    #$00
$2002 LDA    #$41
$2004 STA    $0400,X
$2007 INX
$2008 BNE    $2004
$200A RTS
```

Assembleria käytettäessä huolestamme suhteellisen osoittamisen hankaluudesta ovat tyystin poissa. Assembly-ohjelmaa kirjoitettaessa paikka, johon haaraudutaan, annetaan ikäänkuin se olisi absoluuttinen osoite. Esimerkissä kirjoitimme käskyn BNE \$2004, jossa \$2004 on haarautumispaikan TODELLINEN osoite!

Assembler hoitaa puolestamme raskaan työn ja laskee oikean luvun BNE-käskyn perään. Jos haarautumisosoite sijaitsisi liian kaukana (kauempana kuin 128 tavua eteenpäin tai 127 tavua taaksepäin), assemblerimme antaisi käskyä konekoodiksi kääntäessään virheilmoituksen. Edellisen ohjelmanpätkän assembler sijoittaisi muistiin seuraavassa muodossa:

```
$2000 A2 00    LDX #$00
$2002 A9 41    LDA #$41
$2004 9D 00 04 STA $0400,X
$2007 E8       INX
$2008 D0 FA    BNE #$2004
$200A 60       RTS
```

Huomaa, että antamamme kahden tavun haarautumisosoite \$2004 onkin muuttunut yksitavuiseksi FA:ksi. Mutta minkä takia FA eli varsin suuri luku (250), vaikka peruutimme vain 6 tavun verran?

Selitys löytyy mikroprosessorin sisäisestä tavasta laskea negatiivisilla luvuilla. Sen takia ei tarvitse enemmälti päättää vaivata, kunhan muistaa, että yksi pykälä

taaksepäin on FF eli 255, kaksi pykälää on FE eli 254 jne. Muista, että nollapiste sijaitsee seuraavan käskyn ensimmäisessä tavussa eli esimerkissämme osoitteessa \$200A.

Kirjan lopussa olevasta heksalukutaulukosta voit tarvittaessa aina laskea oikeat argumentit haarautumisoisoiiteisiin, jos koodaat ohjelmaasi käsin ilman assemblerin apua.

Edellisestä esimerkistä löytyi myös vastaus mieltäsi mahdollisesti kaivaneeseen yksityiskohtaan aiempien kappaleiden ohjelmaesimerkeissä. Assembler-listauksissa haarautumiskäskyjen argumenttina on kaksitavuinen luku, mutta ohjelmalaskuria kuvaava oikeanpuolimmainen sarake kasvaa siitä huolimatta vain kahdella. Assembler tietää jo käskyn saadessaan, että sen lopullinen pituus on vain kaksi tavua, joten se osaa kasvattaa ohjelmalaskuria oikealla tavalla.

Hienoimmat assemblerit eivät kaipaakaan osoituksessa edes määränpään absoluuttista osoitetta, vaan ohjelmoitaessa kyseiselle paikalle voidaan antaa nimi, "leima", kirjoittamalla se rivin alkuun. Haarauduttaessa voidaan BNE-käskyn perään sijoittaa tuo annettu nimi, jolloin haarautuminen lasketaan nimettyyn paikkaan.

|       |                  |                                                                            |
|-------|------------------|----------------------------------------------------------------------------|
|       | <b>JMP hyppy</b> | (JMP = Jump eli hyppykäsky. Hypätään käskyn, jonka nimi on 'hyppy')        |
|       | *                |                                                                            |
|       | *                |                                                                            |
|       | *                |                                                                            |
| hyppy | <b>LDA #\$41</b> | (käskyn "nimi" on rivin alussa oleva 'hyppy'.<br>Jatketaan ohjelmaa tästä) |

Haarautumisoisoiiteiden laskentamenetelmä on tunnettava, jos aikoo saada selkoa Basicin DATA-lauseissa tai heksakoodilistana esitetyistä konekieliohjelmista.

Suhteellisella osoitusmuodolla on varsin suuri merkitys ohjelmoinnissa ja varsinkin ohjelmien käytössä. Se mahdollistaa ns. vapaasti sijoitettavien ohjelmien teon. Vapaasti sijoitettava tarkoittaa sitä, että konekielinen ohjelma voidaan sijoittaa periaatteessa minne tahansa muistissa ja se toimii samalla tavalla kuin alkuperäisissäkin osoitteissaan. Jos haluat tehdä vapaasti sijoitettavia ohjelmia, ei ohjelmasi saa viitata omiin osoitteisiinsa suoraan, eli JMP-hyppykäskyt ovat poissa laskuista. Suhteellinen osoitustapa kertoo ainoastaan, kuinka monta käskyä peruutetaan tai ohitetaan, joten se toimii samalla tavalla sijaitse sitten muistin alku- tai loppupäässä (tai missä tahansa sillä välillä).

Huomaat vapaasti sijoitettavien ohjelmien autuuden kouriintuntuvasti ensimmäisen kerran silloin, kun haluat käyttää yhtä aikaa kahta tekemääsi konekielistä rutiinia ja huomaat, että olet sijoittanut ne alunperin molemmat samalle "turvalliselle" alueelle koneesi muistissa. Jos ne ovat vapaasti sijoitettavia, voit

siirrellä niitä vaikka useaankin otteeseen "pois jaloista" ja ne toimivat edelleen halutulla tavalla.

### Haarautuminen eteenpäin

Taaksepäin haarautumiset ovat selväpiirteisiä: haaratumiskäskyä kirjoitettaessa tiedämme jo osoitteen, jonne aiomme käskyn kohdistaa. Mutta entä haarautumiset eteenpäin? Minkä osoitteen me silloin assemblerille annamme tai heksoina laskemme?

Eteenpäin haarauduttaessa ei käsikoodauksessa ole muuta vaihtoehtoa kuin yrittää arvata oikea kohdeosoite ja käyttää sitä. Arvaus on syytä merkitä paperille muistiin tulevaa käyttöä varten. Kun ohjelmanteko on edennyt haarautumisen kohdepaikkaan asti, voimme tarkistaa arvauksemme paikkansa pitävyyden. Jos arvauksemme ei osunut kohdalleen, voimme käydä korjaamassa haarautumiso-soitteen oikeaksi. Eri assembler-versiot toimivat tässä yhteydessä eri tavalla: voimme kirjoittaa koko kyseisen käskyrivin uudestaan tai voimme POKeta suoraan oikeaan paikkaan uuden haarautumisarvon.

Jotkut yksinkertaisimmista assembler-ohjelmista sallivat yhden tai muutaman nimen antamisen eteenpäin suuntautuville haarautumisille. Kehittyneimmät assembler-ohjelmat sallivat rajattoman nimien eli "leimojen" antamisen ohjelmariveille. Näissä tapauksissa sinun ei tarvitse arvata ja korjata, vaan assemblerisi tekee työn kaikessa hiljaisuudessa.

### Sisältyvä (Implied) ja akku (Accumulator)

Nämä kaksi ovat jälleen ryhmiä, jotka eivät oikeastaan ole osoitusmuotoja lainkaan. Niissä toiminnan kohde on joko ilmeinen tai sitten jokin prosessorin sisäisistä rekistereistä, joilla ei osoitteita ole. Näillä käskyillä ei myös ole koskaan argumenttia perässään, ei osoitetta eikä välitöntä lukua. Ne ovat aina täsmälleen yhden tavun mittaisia käskyjä. Sen vuoksi niiden suorituskin tapahtuu nopeammin kuin muiden käskyjen.

Sisältyvään osoitusmuotoon kuuluu mm. käskyt DEX ja DEY, jotka vähentävät X- ja Y-rekisterien arvosta yhden (INX ja INY lisäävät niihin yhden), TAX, joka siirtää akun sisällön X-rekisteriin (TXA toimii päin vastoin) ja SEC, joka asettaa tilarekisterissä muistinumeron eli carryn ykköseksi (CLC nollaa sen). Tähän ryhmään kuuluvat myös RTS, joka palauttaa ohjelman kontrollin pääohjelmaan aliohjelman loppuessa, ja NOP, joka ei suorita mitään tehtävää.

NOP (No OPeration = ei toimintaa) ei todellakaan tee mitään. Ohjelmalaskurin arvoa tietenkin kasvatetaan, kun käsky haetaan, mutta muuten keskusyksikössä on aivan hiljaista. Hämmäntävä käsky, mutta sillekin löytyy käyttöä. Ohjelmaa korjattaessa voi turhaksi tai virheelliseksi osoittautuneen käskyn pyyhkiä yli riittävällä määrällä NOPpeja, jos ei halua jostain syystä kääntää koko ohjelmaa uudestaan. Voit löytää virheellisen käskyn myös pyyhkimällä epäilemäsi

käskyn väliaikaisesti NOPilla ja ajamalla ohjelman sellaisenaan.

Assembler-ohjelmissa on yleensä keinoja varata tilaa taulukoille ja kuvaruudulle välitettävälle merkkijonoille (jotka täytyy sijoittaa muistiin ohjelman keral-la). Käsien koodattaessa saattaa sen sijaan tulla eteen tilanne, jossa muutaman ta-vun haluaisi "pistää sivuun" tulevaa käyttöä varten. Silloin on mukavinta sijoit-taa niihin NOP-koodeja, jotka olemassaolollaan muistuttavat täydennysten tar-peesta.

'Akku' on osoitusmuotona täysin 'sisältyvää' vastaava, mutta tavan mukaan se aina luokitellaan omaksi osoitusmuodokseen. Akku-käskyjä ovat mm. LSR, joka siirtää akun bittejä pykälän verran oikealle. Oikeanpuolimmaisina putoaa akusta pois, mutta se siirretään C-lippuun talteen. Seuraava esimerkki kuvaa LSR-käs-kyn mainiota ominaisuutta jakaa binaariluku kahdella:

```
$0350   LDA #88
$0352   LSR (LSR-käskyllä voi siirrellä myös muistipaikkojen
        bittejä antamalla käskyn perään osoitteen.)
$0353   STA $0360
$0356   RTS
```

Tämän ohjelman suorituksen jälkeen voit PEEKata osoitetta \$0360 ja todeta siellä olevan luvun 68 eli heksana 44 (88/2). Binaariluvun jakautuminen kahtia LSR-käskyllä perustuu siihen, että jokainen ykkösbitti siirtyy käskyn vaikutuk-sesta paikkaan, jossa se merkitsee vain puolta siitä, mitä se merkitsi alkuperäises-sä paikassa. Jos jaettavan luvun oikeanpuolimmaisina eli vähiten merkitsevä bitti on ykkönen (parittomat luvut) ei jako toimi oikein, mutta tämä tilanne on hal-lussa C-lipun avulla, jota testaamalla voidaan ottaa huomioon myös 'puolikkaat' bitit (BCC = haarautuminen, jos C-lippu on nolla, tai BCS = haarautuminen, jos C-lippu on yksi).

## Harkintaa

Tässä olivat 6502:n kaikki 13 osoitusmuotoa. Valinnan vaikeuksia ei kuitenkaan pääse syntymään, sillä kaikki osoitusmuodot eivät ole kerralla käytössä millään käskyllä. Useilla käskyillä on peräti vain yksi vaihtoehto. Usein tarvittava osoi-tusmuoto on kussakin tilanteessa selviö, jota ei tarvitse ohjelmoitaessa ryhtyä miettimään.

Ainoa harkintaa aiheuttava valintatilanne syntyy epäsuorien osoitusten käytös-tä: kannattaako käyttää epäsuoraa vai selvittääkö yksinkertaisemmalla osoituk-sella, esimerkiksi indeksoitu absoluuttinen osoitus (absoluuttinen,X tai absoluut-tinen,Y). Näiden osoitusmuotojen esittelyn yhteydessä läpi käydyt esimerkit jo osoittivat, ettei monimutkaisinta rakennetta kannata aina käyttää, ohjelmasta

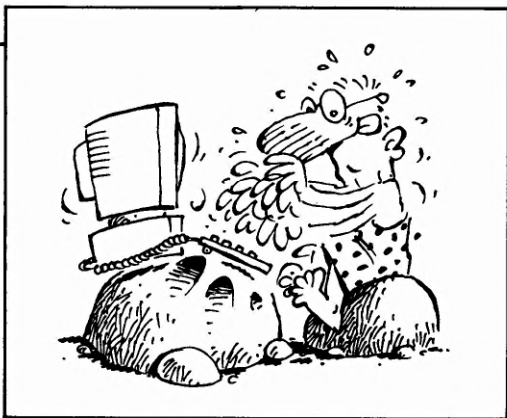
saattaa tulla yllättäen pitempi lukuisten alkuasetusten takia. Jos useampaan kertaan suoritettava ohjelmalohko on pitkä, saattaa epäsuora osoitustapa käydä edulliseksi. Silloin vältetään saman lohkon kirjoittamiselta moneen kertaan.

### YHTEENVETO OSOITUSMUODOISTA

|                  |              |                                                                                 |
|------------------|--------------|---------------------------------------------------------------------------------|
| Välitön:         | LDA #01      | (lataa akkuun luku \$01)                                                        |
| Absoluuttinen:   | LDA \$2000   | (lataa akkuun luku osoitteesta \$2000)                                          |
| Nollasivun:      | LDA \$9F     | (lataa akkuun luku osoitteesta \$9F)                                            |
| Akkurekisteri:   | ASL          | (ASL-käskyllä on muitakin osoitusmuotoja)                                       |
| Sisältyvä:       | DEX          | (vähennä X-rekisterin sisältöä yhdellä)                                         |
| Epäsuora,X:      | LDA (\$FC,X) | (\$FC:hen lisätään X ennen käyttöä epäsuorana osoitteena)                       |
| Epäsuora,Y:      | LDA (\$FC),Y | (\$FC:n ja \$FD:n sisältämään osoitteeseen lisätään Y ennen käyttöä osoitteena) |
| Nollasivu,X:     | LDA \$45,X   | (lataa akkuun luku osoitteesta \$45+X)                                          |
| Nollasivu,Y:     | STX \$45,Y   | (tallenna X-rekisteri osoitteeseen \$45+Y)                                      |
| Absoluuttinen,X: | LDA \$2000,X | (lataa akkuun luku osoitteesta \$2000+X)                                        |
| Absoluuttinen,Y: | LDA \$2000,Y | (lataa akkuun luku osoitteesta \$2000+Y)                                        |
| Epäsuora:        | JMP (\$25E0) | (hyppää osoitteeseen, joka löytyy muistipaikoista \$25E0 ja \$25E1)             |
| Suhteellinen:    | BNE \$0370   | (jos Z-lippu on nolla, hyppää osoitteeseen \$0370)                              |

Suhteellisessa osoitusmuodossa prosessori ei varsinaisesti hyppää suoraan annettuun osoitteeseen, vaan assembler-ohjelma laskee muistiin käskykoodin jälkeen sijoitettavaksi luvuksi sellaisen luvun, joka lisäämällä (hyppy eteenpäin) tai vähentämällä (hyppy taaksepäin) ohjelmalaskurista päästään haluttuun osoitteeseen. Hyppy voi olla enintään 127 tavua eteenpäin tai 128 tavua taaksepäin seuraavan käskyn osoitteesta.

# 6



## Pienet palaset

Commodore 64 ja VIC-20 -tietokoneissa ja monessa muussa käytetään 6502-prosessoria. 64:ssä prosessorin tyyppimerkintä on 6510, mutta toiminnaltaan se on aivan samanlainen kahta ensimmäistä muistipaikkaa lukuunottamatta. Usein sen sanotaan sisältävän 56 erilaista käskyä. Se ei pidä paikkaansa. Käskyjä on itse asiassa 151 kappaletta.

Käskytyyppejä, jotka assembly-kielessä merkitään samalla kirjainlyhenteellä, on tuo 56. Mutta kuten jo olemme huomanneet, piilee esimerkiksi LDA-käskyn eli akun latauskäskyn takana itse asiassa 8 eri käskyä. Kukin niistä lataa akkuun luvun eri tavalla, eri osoitusmuotoa käyttäen.

Kehittyneimmissä Basicissa on myös puolisentoista sataa käskyä. Tasoissa siis tuntuu olevan, vaikkakin Basic-käskyistä löytyy samanlaisia variaatioita kuin osoitusmuodot ovat konekielessä.

Konekielen käskyt ovat yksinkertaisempia aiheuttamiltaan toiminnoilta, mutta usein niissä piilee suoraviivaista voimaa, jota harvoin löytyy Basicista. Konekielellä voidaan syöttää tekstiä ja grafiikkakuvioita suoraan kuvaruudulle ja itse asiassa täyttää koko kuvaruutu silmänräpäyksessä! Teepä se Basicilla.

Keskuşyksikön ymmärtämiä käskyjä on siis 151 kappaletta. Yhdellä tavalla pystyi kuitenkin kuvaamaan 256 erilaista lukua. Miksi sata kappaletta on jäänyt käyttämättä? Tiedä häntä, tyhjiä alucita käskykartasta kyllä löytyy, mutta kun nykyisestäkin käskykannasta suuri osa lojuu tyhjän panttina, olisiko ollut aiheellista niitä enää lisätäköön. Nykyisellä käskykannalla saadaan kaikki oleellinen suoritetuksi.

### Toiminnalliset ryhmät

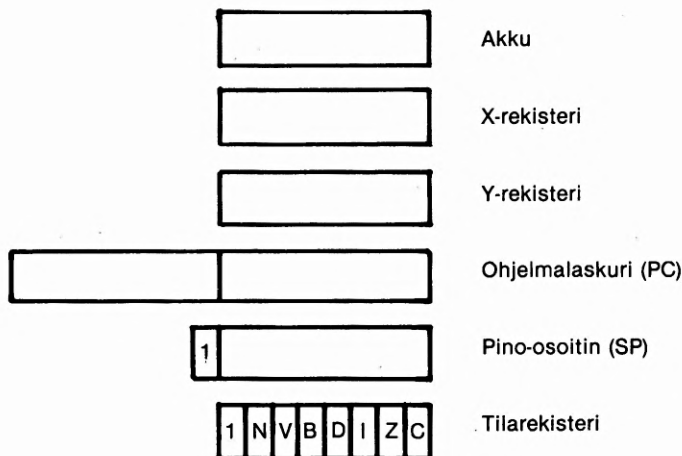
6502:n 56 käskytyyppiä jakautuvat selvästi neljään eri ryhmään toimintansa laadun perusteella: siirtokäskyt, laskentakäskyt, haarautumiskäskyt ja tilarekisteriin liittyvät käskyt. Laskentaryhmässä on muunlaistakin manipulaatiota kuin aritmeettisia toimintoja sisältäviä käskyjä.



# SIIRTOKÄSKYT: LDA ja STA, LDX ja STX, LDY ja STY PHA ja PLA, PHP ja PLP TAX ja TXA, TAY ja TYA, TSX ja TXS

Siirtokäskyt ovat välttämätön perustoiminto tietokoneen keskusyksikössä: lukujen on kunkin vuorollaan päästävä käsiteltäväksi ja toisaalta tieto on pystyttävä kuljettamaan sinne, missä sitä tarvitaan.

Kaikkien rekisterien sisältöä ei voi suoraan siirtää muistiin, mutta toisten rekisterien välityksellä sekin käy.



*6502 keskusyksikön ohjelmointimalli eli käytettävissä olevat rekisterit ja niiden koko. Välittömässä käytössä on akun lisäksi vain kaksi indeksirekisteriä (X ja Y) sekä tilarekisteri. Pino-osoittimen ja ohjelmanaskuri keskusyksikkö ja suoritettava ohjelma hoitavat automaattisesti.*

Ainoa keskusyksikön sisäisistä rekistereistä, johon ei pääse konekielen käskyillä käsiksi on ohjelmanaskuri. Alkuasetuksen jälkeen se hoitaa itse itsensä, tosin ohjelman ohjaamana. Alkuasetuksenkin hoitaa keskusyksikkö.

Ohjelmanaskurin sisältö kasvaa aina yhdellä, kun muistista haetaan käsky. Silloin ohjelmanaskuri osoittaa joko käskyä seuraavaan argumenttiin tai seuraavaan käskyyn, mikäli edellinen käsky oli vain yhden tavun mittainen. Jos käskyyn liitettiin argumentteja, se käy ilmi käskyn koodista ja keskusyksikkö hakee nekin ohjelmanaskurin avustamana kasvattaen sen sisältöä aina yhdellä.



Kun kaikki argumentit (yksi tai kaksi) on haettu, osoittaa ohjelmalaskuri jälleen seuraavaksi suoritettavan käskyn sisältöä.

Haarautumiset ovat ainoa keino muuttaa ohjelmalaskurin sisältöä. Jos ohjelmassamme on käsky JSR (hyppää aliohjelmaan), seuraa sitä kaksi tavua, jotka muodostavat aliohjelmamme alkuosoitteen. Käskyn tuloksena tämä osoite siirretään ohjelmalaskurin sisällöksi. Ennen sitä ohjelmalaskurin edellinen sisältö tallennetaan pinomuistiin pino-osoittimen avulla, jotta se on jälleen käytettävissä palatessamme aliohjelmasta.

## Siirrot prosessorin ja muistin välillä.

Tämän ryhmän L-kirjaimella alkavat käskyt lataavat eli hakevat aina luvun jostain muistipaikasta johonkin prosessorin sisäiseen rekisteriin. Vastaavasti S-kirjaimella alkavat käskyt tallentavat luvun jostain rekisteristä muistialueelle.

LDA ja STA ovat vanhat tuttumme edellisistä luvuista. Akku on varsin paljon käytetty prosessorin rekisteri ja sen takia siirrot akun ja muistin välillä ovat tärkeässä asemassa. LDA-käskyä on olemassa kahdeksan erilaista, joista jokainen lataa luvun akkuun eri tavalla, eri osoitusmuotoa käyttäen.

Haluttu osoitustapa ilmaistaan assembleria käytettäessä kirjoittamalla käskyn perässä oleva osoiteosa eri tavoin.

Akku on rekisteri, jonne tietoa tuodaan, jossa sitä käsitellään ja josta se taas seuraavassa silmänräpäyksessä viedään johonkin toisaalle muistissa.

STA-käskyssä osoitusmuotoja on vain seitsemän. Ne ovat samat kuin LDA-käskylläkin. Puuttuva osoitusmuoto on "välitön", jolla LDA-käskyn yhteydessä ladattiin akkuun välittömästi käskykoodia seuraava luku itse ohjelman sisällä. STA-käskyn yhteydessä se merkitsi argumenttina olevan luvun sijoittamista johonkin muistipaikkaan.

Vaikkakin se olisi erittäin käyttökelpoinen käsky, ei sitä kuitenkaan ole 6502:ssa mukana. Se nimittäin vaatisi pitemmän käskyn kuin on mahdollista: sioitettava luku veisi yhden tavun ja osoite tämän perässä vielä kaksi tavua.

Joka tapauksessa assemblerilla ohjelmoitaessa on syytä muistaa, että välitöntä osoitusmuotoa ei STA-käskyn yhteydessä ole.

Käskyt LDX ja LDY lataavat vastaavasti 6502:n rekistereihin X ja Y jonkin luvun. Indeksirekistereitä X ja Y käytetään etupäässä ohjelmasilmukoiden muodostamiseen. Niihin ladataan jokin luku, suoritetaan ohjelmanpätkä ja vähennetään kyseisen indeksirekisterin arvoa yhdellä. Ohjelmanpätkä suoritetaan sen jälkeen uudestaan hyppäämällä ohjelmassa takaisin päin. Indeksirekisterin arvoa vähennetään jälleen yhdellä, kunnes se saavuttaa arvon nolla. Indeksirekisterin muuttuminen nolaksi testataan tietyllä käskyllä, jolloin jatketaan ohjelman suoritusta taas eteenpäin.

Indeksirekistereitä käytetään myös ohjelmasilmukoiden sisällä liu'uttamaan osoitearvoa halutun muistialueen yli. Tämähän tapahtuu vaikkapa osoitusmuo-

dolla 'absoluuttinen,X', esimerkiksi LDA \$0400,X. Tässä käskyssä osoitteeseen \$0400 lisätään X-rekisterissä oleva luku ja vasta siitä osoitteesta haetaan luku akkuun. Muuttamalla X:n arvoa ohjelasilmukassa saadaan akkuun ladattua peräkkäisillä kerroilla luvut peräkkäisistä muistipaikoista.

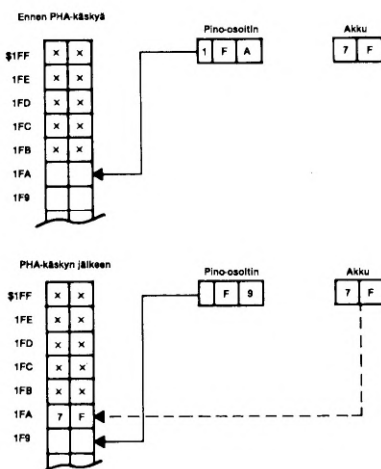
## Pinon käyttö

P-kirjaimella alkavat tämän ryhmän käskyt koskettelevat kaikki muistisivulla kaksi sijaitsevaa pinoa. P-kirjain on helppo muistaa Pinosta, vaikkei se tästä suomen kielisestä sanasta olekaan lyhennetty.

Keskusyksikössä sijaitsee rekisteri, jonka nimi on pino-osoitin. Tässä vaiheessa on tehtävä selvä ero pino-osoittimen ja itse pinon välillä: pino-osoitin on 6502:n sisällä sijaitseva yhden tavun mittainen rekisteri, jota KÄYTETÄÄN OSOITTEENA, kun pinolle halutaan tehdä jotain. Itse pino sijaitsee muistialueella (sivulla kaksi, osoitteet \$100 — \$1FF).

Pino-osoitin osoittaa aina johonkin muistipaikkaan tällä alueella. Kun pino-osoitinta käytetään osoitteena, lisätään siihen yhdeksäs bitti, joka on aina ykkönen. Sillä tavalla pino pystyy osoittamaan muistin toiselle sivulle eli sivulle 1 (ensimmäinen sivu on sivu 0), vaikka se on vain 8 bitin kokoinen.

Kun haluamme siirtää jonkin luvun pinoon, siirretään se pino-osoittimen osoittamaan muistipaikkaan. Samalla pino-osoittimen sisältöä vähennetään yhdellä. Pino-osoitin osoittaa siis aina seuraavaa vapaata paikkaa. Tallennettava lukujono muodostuu pinossa ikään kuin ylhäältä alaspäin. Käynnistettäessä tietokonetta pino-osoittimen arvoksi sijoitetaan \$FF (255), johon sijoitetaan ensimmäinen pinoon tallennettava luku. Seuraavat luvut sijoittuvat aina pienempiin ja pienempiin muistiosoitteisiin.



PHA-käsky kopioi akun sisällön pino-osoittimen osoittamaan paikkaan pino-muistialueella. Osoittimen sisältöä vähennetään yhdellä. PHA-käsky toimii toisin päin: ensin osoittimen sisältöä kasvatetaan yhdellä ja sen jälkeen akkuun kopioidaan osoittimen osoittaman muistipaikan sisältö. Aliohjelmaan hypättäessä pinoon tallennetaan sen hetkinen ohjelasikuriin arvo, jotta aliohjelmaasta voitaisiin palata samaan paikkaan takaisin. Ohjelasikuri on kaksi tavua pitkä, joten ensin tallennetaan osoitteen enemmän merkitsevä tavu ja sen jälkeen osoitteen vähemmän merkitsevä tavu. RTS-käskyyn törmättäessä pinossa oleva paluuoite sijoitetaan takaisin ohjelasikuriin.

Kun pino-osoitin sisältää luvun \$00, eli olemme tallentaneet pinoon 255 lukua ottamatta yhtäkään pois, on parasta jättää pino rauhaan. Jos vielä käyttäisimme pinoa, lipsahtaisi se lukualueensa yli ja osoittaisi seuraavan kerran jälleen lukua \$FF, jossa meillä jo oli tallennettuna ensimmäinen lukumme ja kirjoittaisimme sen päälle.

PHA vie pinoon akun sisältämän luvun ja PLA hakee akkuun luvun pinosta. PHP ja PLP ovat vastaavat käskyt tilarekisterille.

Osoitteesta ei pinoa käytettäessä tarvitse välittää, keskusyksikkö hoitaa sen automaattisesti, minkä takia pinokäskyt ovatkin lyhyitä ja nopeita.

Tässä piilee houkutus käyttää pinoa kaikenlaisen välitulosten tallentamiseen, mutta muista: pinoa käyttää myös muut lohkot tietokoneesi sisällä (keskeytysohjelmat, aliohjelmakutsut), joten sinun käytettävissäsi ei ole koko 256 tavun tilaa!

Toinen muistettava asia: Pino on ehdottomasti pidettävä puhtaana! Toisin sanoen: jos tallennat sinne jotain, on se ehdottomasti otettava sieltä myös pois!

Aliohjelmiin hypättäessä paluuosoitteet tallennetaan pinoon ja olisi hienoa harmilista, jos aliohjelmasta palattaessa pinoon olisi jäänyt sinun jättämiäsi lukuja, joita keskusyksikkö yrittäisi tulkita oikeiksi paluuosoitteiksi!

Muista, että pinoa käyttävät käskyt kohdistuvat muistiin, ei itse pino-osoittimeen.

## Sisäiset siirrot

Keskusyksikön sisäisten rekisterien välillä lukuja voidaan siirtää varsin vapaasti. Ainoa poikkeus on pino-osoitin, joka voi "keskustella" vain X-rekisterin kanssa. Muunlaiseen manipulointiin pino-osoittimen kanssa ei ole tarvettakaan, koska lähes ainoa tilanne, jolloin siihen puututaan on tietokoneen käynnistys virran kytkemisen jälkeen. Silloin pino-osoittimen "siemenluvuksi" annetaan X-rekisterin välityksellä \$FF ja sen jälkeen se hoitaa itse itsensä.

Sisäiset siirrot alkavat kaikki T-kirjaimella (englanniksi Transfer = siirrä). TAX ja TXA siirtävät luvun akusta X-rekisteriin ja päinvastoin. TAY ja TYA tekevät saman akun ja Y-rekisterin välillä. TXS siirtää luvun X-rekisteristä pino-osoittimeen ja TSX pino-osoittimesta X-rekisteriin. Tällä käskyllä voidaan tarkistaa, onko pino jo täynnä, jos sinne halutaan tallentaa jotain.

Lukujen siirrossa on itse asiassa kyse kopioinnista. Jos siirrämme TAY:llä luvun akusta Y-rekisteriin, säilyy akun sisältö edelleen ennallaan.

## Tilarekisterin liput

Osalla siirtokäskyistä on muunkinlaista vaikutusta kuin lukujen siirto. Rekisterien lataukset ja sisäiset rekisterisiirrot asettavat nimittäin tilarekisterissä olevat negatiivi- ja nollaliput, jos siihen on aihetta.

Jos lataamme akkuun luvun ja se osoittautuu nollaksi, asettuu tilarekisterin Z- eli nollalippu ykköseksi. Tämän avulla välitetään seuraavalle käskylle tieto siitä,

että ladattu luku oli nolla.

Keskusyksiköllä ei ole näkökykyä, joten sen on käytettävä toiminnassaan tällaista "sormituntumaa" nostamalla ja laskemalla tilarekisterin lippuja eli muutamalla tietty bitti ykköseksi tai nollassi. Jos ladattu luku ei ole nolla, on latauksen jälkeen Z-lipun arvo nolla.

Negatiivi- eli N-lippu asettuu ladattaessa ykköseksi, jos ladattavan luvun eniten merkitsevä bitti (vasemmanpuoleisin eli bitti 7) on ykkönen. Tällä on merkitystä niissä tapauksissa, jolloin ohjelma käsittelee tavun sisältämää tietoa sekä negatiivisena että positiivisena sen mukaan, onko bitti 7 ykkönen vai nolla. Tällä tavalla merkittäessä lukualue on -128 ... +127. Etumerkillisen aritmetiikan käyttö vaatii kuitenkin varsin hyvää ohjelmointitaitoa eikä sitä tarvita juuri lainkaan tavallisimmissa konekieliohjelmissa.

N-lipulle meillä on tässä kirjassa vain hyvin vähän käyttöä. Kaikki, mitä siitä tarvitsee tietää, on että se asettuu, jos ladattavan luvun merkitsevin bitti on ykkönen.

Meitä kiinnostavat ja tiheään käytetyt liput ovat C-lippu eli muistinnumero ja Z- eli nollalippu. Näiden lippujen sisältämiä tietoja käytetään ohjelmoinnissa hyvin usein hyväksi.

Konekielisessä ohjelmassa suoritetaan IF...THEN -tyyppistä päättelyä ainoastaan lippujen perusteella. Näihin lippuihin palataan vielä haarautumiskäskyjen yhteydessä perusteellisemmin.

STA-tyyppiset käskyt eivät vaikuta lippujen sisältöön lainkaan. Se ei ole tarpeenkaan. Jos voimme tallentaa jonkin rekisterin arvon muistiin, olemme sen myös jostain ladanneet, ja silloin asiaankuuluvat liput ovat jo asettuneet. Toisaalta olemme ehkä tehneet jonkin operaation, esimerkiksi yhteenlaskun, joka on asettanut liput ja haluamme siirtää luvun takaisin muistiin, ennen kuin teemme johtopäätöksiä lippujen tilasta. Silloin on tärkeää, ettei tallennuskäsky sekoita lippujen sisältöä.

Pinoon siirrot ja X-rekisterin siirto pino-osoittimeen eivät myöskään muuta lippujen tilaa. Kirjan lopussa olevassa käskyluettelossa on tarkkaan selvitetty käskyn vaikutus eri lippuihin. Jos olet ohjelmoidessasi vähääkään epävarma käskyjen vaikutuksesta, voit tarkistaa asian taulukosta.

Haarautumisia aiheuttavat käskyt ovat itse asiassa tietokoneen äly. Niiden avulla tietokone tekee johtopäätöksiä ulkoisen maailman tapahtumien ja lukujen sisältöarvojen perusteella. Ilman haarautumisia ei tietokone voisi valita useamman toimintavaihtoehdon väliltä.

Tilarekisteriin kohdistuvat käskyt ovat apuna laskentaa ja haarautumisia suorittavien käskyjen toiminnassa. Ne toimivat keskusyksikön muistiinpanovälineinä. Ne merkkäävät muistiin tiettäjä tosiasioita seuraavien käskyjen tutkittaviksi. Tilarekisteriin vaikuttavat myös monet muut, etenkin laskentakäskyt, automaattisesti.

## LASKENTA JA LOGIIKKA:

**ADC ja SBC; AND, EOR ja ORA  
BIT, CMP, CPX ja CPY  
ASL, LSR, ROL ja ROR  
DEC, DEX, DEY, INC, INX ja INY**

Laskennalliset ja muulla tavalla tavujen sisältämiä lukuja muokkaavat käskyt muodostavat perustan koko tietokoneen älykkäältä näyttävälle toiminnalle. Ilman niitä tietokone olisi kuin puhelinkeskus: tieto siirtyy vikkelaan, mutta pysyy aivan alkuperäisen kaltaisena.

Laskenta on eniten harmaita hiuksia tuottava toimenpide Basiciin tottuneelle konekieliohjelmoijalle. Konekielellä et voi tehdä paljoakaan: yhdellä käskyllä (ADC) voit laskea lukuja yhteen, mutta luvut eivät saa olla suurempia kuin 255. Yhdellä käskyllä (SBC) voit myös vähentää kaksi lukua toisistaan, mutta sama rajoitus pätee. Se ei ole paljoakaan, kun vertaa Basiciin: kokonaislukualue -32768...+32767, joissakin Basicissa vieläkin suurempi; lisäksi vielä liukulukuaritmetiikka ja yleensä vähintään kuuden merkitsevän numeron tarkkuudella kaikki matemaattiset funktiot, mitä ajatella saattaa. Kaikki tämä yhden käskyn puitteissa. Mahdollisuuksia Basicissa on, mutta se maksaa myös: ohjelmat ovat hitaita.

Konekielessä saat tyytyä yhden tavun laskutoimituksiin, joten tuskin kannattaa konekielellä tehdä apuohjelmia kotitehtävien ratkaisemiseen! Eipä silti, ei niitä tarvitsekaan. Basic on matematiikkaa sisältävissä ohjelmissa aivan omiaan. Basicin nopeus riittää aivan hyvin kertomaan sinulle laskujesi vastaukset. Konekieli on paikallaan nopeata grafiikkaa ja monimutkaista ääntä ohjelmoitaessa sekä erikoisefektiön luonnissa. Niissä se pystyy uskomattomiin suorituksiin. Jättäkäämme suosiolla matemaattiset tehtävät korkeamman tason kielille.

Konekielellä joudutaan kuitenkin joskus laskemaan isommilla luvuilla kuin 255. Voimme esimerkiksi tarvita peliimme pistelaskurin, joka juoksee nolasta 10 000:een.

Näihin tapauksiin löytyy yksinkertaiset lääkkeet. Niputamme muutaman tavun muistissa peräkkäin ja käsittelemme niitä kuin ne olisivat joko 16-bittinen luku (2 tavua) tai 24-bittinen luku (3 tavua). Lukualueet ovat niillä vastaavasti 0...65535 ja 0...1677215. Näillä pärjää jo pitkälle ja mikään ei estä ottamasta käyttöön vielä useamman tavun pituisia lukuja. Konekielellä kaikki on mahdollista.

## Yhteenlasku

ADC on käsky, jolla lasketaan kaksi lukua yhteen. Toinen yhteenlaskettavista sijaitsee aina akussa, eli se on ladattava sinne ennen ADC-käskyä. Käskyllä on osoitusmuotoja yhtä monta kuin LDA:llakin, eli akkuun lisättävä luku voi sijaita joko välittömästi seuraavassa ohjelmatavussa tai sitten jossain muistipaikassa.

X- ja Y-rekistereihin ei voi kohdistaa yhteenlaskuoperaatiota, vastaavia käskyjä ei valikoimasta löydy.

ADC on lyhenne englannin kielen sanoista ADd with Carry eli lisää muistinumeron kera. Yhteenlaskettavien lukujen lisäksi tuloksessa näkyy siis myös muistinumero. Jos muistinumero eli C-lippu oli 1, on tulos yhtä liian suuri, mikäli halusimme laskea vain kaksi lukua yhteen. Ennen ADC-käskyä on siis muistettava toinenkin asia: nollaa C-lippu käskyllä CLC (CLear Carry = nollaa muistinumero).

Syy siihen, että C-lippu kummittelee yhteenlaskussa mukana, löytyy useamman tavun yhteenlaskuista. Jos olemme ajatelleet käyttää kahta peräkkäistä tavua kuvaamaan suurempaa lukua kuin yhteen tavuun mahtuu, on meidän suoritettava yhteenlasku kahdessa osassa: ensin laskemme yhteen vähemmän merkitsevät tavut, sijoitamme tuloksen paikoilleen, sen jälkeen laskemme yhteen enemmän merkitsevät tavut.

Tulos ei ole oikea, elleimme ota huomioon alempien tavujen yhteenlaskusta mahdollisesti tulevaa muistinumeroa. Laskemalla ylemmät tavut yhteen C-lipun kera, siirtyy muistinumero automaattisesti mukaan ylempien tavujen yhteenlaskuun. Tämä on kätevä ominaisuus, mutta sen vuoksi meidän on nollattava muistinumero aina ennen ENSIMMÄISTÄ yhteenlaskua.

Katsotaan esimerkein, mitä tapahtuu, kun laskemme kaksi 8-bittistä lukua yhteen:

Akku ja muistinumero: 

|   |   |      |      |      |      |      |      |      |      |
|---|---|------|------|------|------|------|------|------|------|
| C | — | bit7 | bit6 | bit5 | bit4 | bit3 | bit2 | bit1 | bit0 |
|---|---|------|------|------|------|------|------|------|------|

**Yhteenlasku (ei muistinumeroa):**

|     |       |                                     |
|-----|-------|-------------------------------------|
| CLC |       | (nollaa muistinumero)               |
| LDA | #\$75 | (lataa akkuun 117 eli heksana \$75) |
| ADC | #\$3A | (lisää akkuun 58 eli heksana \$3A)  |

|                     |          |              |        |
|---------------------|----------|--------------|--------|
|                     | 10-järj. | bitteinä     |        |
| akun sisältö:       | 117      | 01110101     |        |
| lisättävä luku:     | + 58     | + 00111010   |        |
| akkuun jäävä tulos: | 175      | C:0 10101111 | (=175) |

**Yhteenlasku 2-järjestelmässä:**

|                                  |
|----------------------------------|
| 0+0=0                            |
| 0+1=1                            |
| 1+0=1                            |
| 1+1=0 (lisäksi muistinumero = 1) |

**Yhteenlasku (muistinumero syntyy):**

|            |            |                                     |
|------------|------------|-------------------------------------|
| <b>CLC</b> |            | (nollaa muistinumero)               |
| <b>LDA</b> | <b>#D5</b> | (lataa akkuun 213 eli heksana \$D5) |
| <b>ADC</b> | <b>#6C</b> | (lisää akkuun 108 eli heksana \$6C) |

|                     |            |              |
|---------------------|------------|--------------|
|                     | 10-järj.   | bitteinä     |
| akun sisältö:       | 213        | 11010101     |
| lisättävä luku:     | + 108      | + 01101100   |
| akkuun jäävä tulos: | 321        | C:1 01000001 |
|                     | 256 + 65 = | 321          |

Laskettaessa yhteen lukuja, joiden summa jää pienemmäksi kuin 256, ohjelmamme eteneminen on mutkatonta. Akkuun jäävä luku on täsmälleen haluamamme summa. Esimerkissä on myös esimerkkilistaus, jolla yhteenlaskutoiminto saadaan aikaiseksi. Huomaa ensimmäisenä käskynä oleva CLC, joka nollaa mahdollisesti asettuneena olevan muistinumeron.

Esimerkin perässä on kerrattu, miten kaksijärjestelmän numeroita lasketaan yhteen. 1+1 on kaksijärjestelmässä 10, joista numeroista ensimmäinen on seuraavaan sarakkeeseen siirtyvä muistinumero.

Akun kaikkien 8 bitin yhteenlasku toisen luvun kanssa sujuu aivan tavanomaisesti ja laskettaessa viimeisiä bittejä (vasemmanpuolimmaisia) yhteen, saadaan vain 1 ilman muistinumeroa. C-lippuun ei siis tule mitään.

Toisessa esimerkissä lukujen summa on yli 255. Sellainen luku ei mahdu akkuun, koska 8 bitillä pystyy esittämään vain luvut 0...255. Silti yhteenlasku sujuu keskusyksikössä kitkattomasti. Bitti bitiltä edetään oikealta vasemmalle (itse laskennan suorittaa prosessorin aritmeettis-looginen yksikkö, ALU).

Viimeisten bittien yhteenlaskussa tulee tällä kertaa kuitenkin vielä muistinumero, joka ei enää mahdu akkuun. Se sijoitetaan C-lippuun osoittamaan ohjelman jatkokäskyille, että akussa oleva luku ei enää olekaan koko tulos.

Tässä tilanteessa voit käyttää C-lippua hyväksesi saadaksesi ohjelmasi toimimaan kunnolla. Jos tarkoituksesi on pitää luvut yhteen tavuun sopivina, et tietenkään voi hyväksyä akkuun jäävää väärää tulosta. Kirjoita ohjelmaasi yhteenlaskukäskyn perään seuraavat käskyt:

|                   |              |                                                  |
|-------------------|--------------|--------------------------------------------------|
| <b>\$0350 BCC</b> | <b>0354</b>  | (BCC: jos C=0, hyppää pari tavua eteenpäin)      |
| <b>\$0352 LDA</b> | <b>#\$FF</b> | (jos C oli 1, korvaa väärä tulos luvulla 255)    |
| <b>\$0354 STA</b> | <b>...</b>   | (talleta yhteenlaskun tulos haluamaasi paikkaan) |

Nämä käskyt aiheuttavat sen, että akun sisältö ei voi kasvaa yli äyräidensä, vaan sen sisältö "tapittaa" 255:een, mikäli yhteenlaskun tulos ylittää sallitun arvon.

Toinen vaihtoehto on korvata käsky LDA #\$FF hypyllä aliohjelmaan, joka



ilmoittaa esimerkiksi kuvaruudulle laskutoimituksessa tapahtuneesta ylivuodosta.

C-lippu toimii yhteenlaskussa ikään kuin akkurekisterin 9. bittinä. Yhdeksännen bitin arvo olisi kymmenjärjestelmässä 256, ja jos siihen lisätään akkuun jäävä tulos 65, saadaan aivan oikein 321.

## Kahden tavun yhteenlasku

Tarkastellaan tapausta, jossa lasketaan yhteen kaksi kahden tavun mittaista lukua. Olemme varanneet muistista paikat \$FB ja \$FC kahden tavun mittaiselle luvulle siten, että osoitteessa \$FB on luvun vähemmän merkitsevä tavu ja osoitteessa \$FC on enemmän merkitsevä tavu. Tavujen järjestys on näin sama kuin se tapa, jolla 6502 haluaa ladata osoitetiedot.

Tällä tavalla muodostettua kahden tavun mittaista lukua voisi siis käyttää epäsuorassa osoitusmuodossa lopullisen osoitteen tallentamiseen, ja sitä voisi käyttää hyväksi esimerkiksi käskyllä LDA (\$FB),Y. Se hakisi akkuun luvun laskemattamme osoitteesta lisättynä Y-rekisterin arvolla.

Oletetaan, että \$FB:ssä on luku \$FE ja muistipaikassa \$FC on luku \$04. Nämä tavut sisältävät siis heksaluvun \$04FE. Haluamme lisätä tähän kahden tavun mittaiseen lukuun heksaluvun \$000F. Käytännössä tämä tarkoittaa sitä, että meillä on muistipaikoissa \$FB ja \$FC osoite, joka osoittaa kuvaruutumuistiin jonnekin 7. rivin tienoille (Commodore 64) ja haluamme siirtää "omaa kursoriamme" eli tätä osoitetta 15 paikkaa eteenpäin (\$000F).

### Kahden tavun yhteenlasku:

|           |                                                                                                                  |
|-----------|------------------------------------------------------------------------------------------------------------------|
| CLC       | (nollataan muistinumero)                                                                                         |
| LDA \$FB  | (nollasivun osoitus! Ladataan akkuun vähemmän merkitsevä tavu eli esimerkkinme tapauksessa \$FE)                 |
| ADC #\$0F | (lisätään akkuun luku \$0F)                                                                                      |
| STA \$FB  | (viedään tulos takaisin muistipaikkaan \$FB)                                                                     |
| LDA \$FC  | (ladataan akkuun toinen tavu. Huomaa, että STA ja LDA eivät vaikuta yhteenlaskussa syntyneeseen muistinumeroon!) |
| ADC #\$00 | (lisätään akussa olevaan toiseen tavuun \$00 JA edellisessä yhteenlaskussa syntynyt muistinumero)                |
| STA \$FC  | (viedään tulos takaisin paikalleen)                                                                              |

|                       |     |          |         |
|-----------------------|-----|----------|---------|
|                       |     | \$FB     |         |
| muistinumerot:        |     | 111111   |         |
| ennen yhteenlaskua:   |     | 11111110 |         |
| lisättävä luku:       | C:0 | 00001111 |         |
| yhteenlaskun jälkeen: | C:1 | 00001101 | (=\$0D) |



|                       | \$FC              |                                    |
|-----------------------|-------------------|------------------------------------|
| muistinumerot:        | 1                 | 1 (edellisen tavun yhteenlaskusta) |
| ennen yhteenlaskua:   | 00000100          |                                    |
| lisättävä luku:       | 00000000          |                                    |
| yhteenlaskun jälkeen: | 00000101 (= \$05) |                                    |

Vastaavalla tavalla voimme laskea yhteen vaikka kuinka monen tavun mittaisia lukuja. Aloitamme vähiten merkitsevästä tavusta ja etenemme tavu tavulta eteenpäin. C-lippu nollataan ennen ensimmäistä yhteenlaskua. Tavuista syntyvät muistinumerot siirtyvät automaattisesti C-lipun välityksellä eteenpäin, mikäli käytämme yhteenlaskujen välillä ainoastaan LDA ja STA -käskyjä. Kirjan lopussa olevasta käskytaulukosta voit katsoa, mitkä ovat muita ”turvallisista” käskyjä.

## Yhteenlasku kymmenjärjestelmässä

6502:lla voidaan suorittaa yhteenlaskua myös siten, että akussa oleva luku tulkitaan ns. BCD-luvuksi. Se tulee sanoista Binary Coded Decimal eli suomeksi binäärisesti koodattu 10-järjestelmä. BCD-luvut muistuttavat heksadesimaalisia lukuja, mutta on varottava sekoittamasta niitä toisiinsa.

BCD-lukuja käsiteltäessä akku jaetaan kahteen neljän bitin ryhmään, aivan kuten akun sisältö tulkitaan heksoinakin. Mutta sen sijaan, että kumpikin neljän bitin muodostama luku voisi olla mitä tahansa välillä 0000 — 1111 (heksoina \$0 — \$F), se voikin kasvaa 0000:sta ainoastaan 1001:een (eli 0...9).

Kun yhteenlaskettaessa neljän vähemmän merkitsevän bitin tulos ylittäisi tuon 1001:n eli 9:n, siirretään tavun sisäinen muistinumero neljälle enemmän merkitsevälle bitille ja nämä vähemmän merkitsevät bitit alkavat ”pyöriä” jälleen nollasta.

Tätä toimintoa ohjaa 6502:ssa tilarekisterin D-lippu. Jos D-lippu on nolla, suoritetaan yhteenlasku aivan normaalisti, mutta jos se on ykkönen, tulkitaan akun sisältö BCD-luvuksi ja yhteenlasku suoritetaan BCD-laskusäännöillä.

Todennäköisesti et ohjelmoidessasi tarvitse BCD-lukujen yhteenlaskua, joten voit tässä vaiheessa vielä unohtaa koko asian. Tärkeätä on kuitenkin muistaa, että D-lippu on syytä ohjelman alkuvaiheessa aina nollata, jottei mahdollisesti voimassa oleva BCD-tila jäisi ohjelmassasi päälle.

## Vähennyslasku

Aivan yhteenlaskua vastaavalla tavalla voit vähentää akkurekisterin sisällöstä jonkin luvun (SBC: SuBtract with Carry = vähennä muistinumeron kera). SBC vähentää akun sisällöstä halutun vakioluvun (välitön osoitus) tai jonkin muistipaikan sisältämän luvun.

Jos vähennyslaskun tulos ei mene pienemmäksi kuin nolla, on kaikki hyvin ja akussa on lukujen oikea erotus. Jos sen sijaan vähennettävä luku on suurempi

kuin akussa oleva luku, joudutaan 'negatiiviselle' puolelle ja ohjelmassa on otettava tämä huomioon.

Vähennyslaskussa ei bitiltä toiselle etene muistinumeroa, kuten yhteenlaskussa, vaan sitä kutsutaan "lainaukseksi". Jos akussa on nolla (\$00) ja vähennämme siitä yhden, tapahtuu seuraavaa: akun sisältö "pyörähtää ympäri" lukuun \$FF ja viimeisten bittien vähennysoperaatiosta syntyy lainausnumero, joka ohjataan C-lippuun.

Päinvastoin kuin yhteenlaskussa, nyt C-lipun nollatila osoittaa ylivuodon tapahtuneen. C-lippua testaamalla voimme jälleen tarkistaa, onko akussa oikea tulos, vai päädyimmekö vähennyslaskun tuloksena "negatiiviselle" alueelle.

Esimerkiksi seuraavan esimerkin lisäkäskyillä voimme rajata vähennyslaskun tuloksen vähintään nollassi:

|        |            |                                                                       |
|--------|------------|-----------------------------------------------------------------------|
| \$2000 | SEC        | (vähennyslaskussa on C-lippu asetettava ensin)                        |
| \$2001 | LDA \$4000 | (ladataan akkuun luku muistipaikasta \$4000)                          |
| \$2004 | SBC #\$3F  | (vähennetään siitä luku \$3F)                                         |
| \$2006 | BCS \$200A | (jos C-lippu on 1, on kaikki hyvin ja hyppäämme seuraavan käskyn yli) |
| \$2008 | LDA #\$00  | (mikäli C-lippu oli 0, "tapitamme" tuloksen nolnaan)                  |
| \$200A | STA \$4000 | tallennetaan tulos takaisin alkuperäiselle paikalle)                  |

Useamman tavun mittaisten lukujen vähennyslaskussa noudatetaan samaa loogikkaa kuin yhteenlaskussa. Vähiten merkitsevät tavut vähennetään toisistaan ensin, sitten seuraavat jne. C-lippu vie tarvittaessa lainausnumeron seuraaville tavuille.

Jos viimeisten tavujen vähennyslaskun jälkeen C-lippu on nolla, meni koko luku "negatiiviselle" ja ohjelman on tavalla tai toisella korjattava tilanne.

## Loogiset operaatiot

Yhteen- ja vähennyslaskun lisäksi akun sisältöä voidaan muuttaa myös AND, EOR ja ORA -käskyillä. Ne eivät suorita laskutoimituksia akun sisältämällä LUVULLA vaan käsittelevät erikseen akkurekisterin sisältämiä BITTEJÄ.

AND on suomeksi JA. Se vertaa akun kutakin bittiä muistista haetun luvun vastaaviin bitteihin. Mikäli molemmat ovat ykkösiä, laitetaan akkuun vastaavalle paikalle ykkönen. Jos jommassa kummassa bitti on nolla, laitetaan akkuun sen bitin paikalle nolla.

AND: 0 JA 0 = 0  
 0 JA 1 = 0  
 1 JA 0 = 0  
 1 JA 1 = 1

akun sisältö: 11110000  
 vertailuluku: 00100010  
 akkuun jäävä luku: 00100000

AND-käsky on kätevä silloin, kun haluamme varmistaa, että akussa tai muistissa olevan tavun jokin bitti on nolla.

Jos esimerkiksi haluamme varmistaa, että muistipaikassa \$4F2A kolmas bitti eli bitti 2 on nolla, haemme kyseisestä muistipaikasta tavun akkuun ja suoritamme sille AND-operaation sopivan luvun kanssa. Jos tämä 'sopiva' lukumme sisältää ykkösen kaikkialla muualla, paitsi bitti 2:n kohdalla pysyvät ne totuustaulukon mukaan alkuperäisissä arvoissaan. Jos bitti 2:n kohdalla siinä on nolla, nollautuu akun bitti 2.

\$0350 LDA \$4F2A (ladataan akkuun bitit osoitteesta \$4F2A)  
 \$0353 AND #\$FB (verrataan akkua lukuun \$FB, joka kaksijärjestelmän luku on 11111011)  
 \$0355 STA \$4F2A (tallennetaan luku takaisin paikalleen. Sen kolmas bitti eli bitti 2 on nyt nolla ja muut bitit ovat säilyneet ennallaan.)

Esimerkissä ANDasimme akun binaariluvun 11111011 kanssa. AND-käsky toimi eräänlaisena 'maskina'. Halusimme puuttua vain akussa olevan luvun kolmanteen bittiin, joten sijoitimme vertailulukuun muiden bittien kohdalle ykkösen, jolloin ne säilyivät operaatiossa ennallaan.

ANDilla voidaan akussa olevasta luvusta nollata mitkä tahansa bitit. Ainoaksi ongelmaksi jää laskea mikä kymmenjärjestelmän tai heksajärjestelmän luku vastaa haluamaamme maskia, esimerkiksi 01010101. Kappaletta 2 kertaamalla muistat, että jokaista ykköstä vastaa tietty luku kymmenjärjestelmässä ja laskemalla ne yhteen saat koko tavua vastaavan kymmenjärjestelmän luvun.  $01010101$  olisi  $64+16+4+1$  eli 85.

AND-käsky ei konekielessä vastaa aivan samaa kuin Basicissa. Basicin AND testaa myös totuusarvoja kokonaisuuksina, esimerkiksi

**300 IF A=1 AND B>5 THEN 340**

Basicin ANDilla voit tehdä myös saman kuin konekielessä, mutta suoraan kymmenjärjestelmän luvuilla, esimerkiksi PRINT 5 AND 3, joka antaisi tulokseksi 1 ( $101 \text{ AND } 011 = 001$ ).

AND-käskyn lähisukulainen on ORA. OR on suomeksi TAI. ORA tutkii myös akun ja muistista haetun (tai välittömästi käskyä seuraavan) luvun yksittäisiä bittejä. Kahden bitin vertailussa tulokseksi tulee kuitenkin yksi, jos toinen TAI toinen bitti on ykkönen.

**ORA: 0 TAI 0 = 0**

**0 TAI 1 = 1**

**1 TAI 0 = 1**

**1 TAI 1 = 1**

**akun sisältö: 00001111**

**vertailuluku: 00110011**

**akkuun jäävä luku: 00111111**

ORA-käskyn argumentti toimii myös eräänlaisena maskina. Nyt maskibitteinä käytetään ykkösiä ja niiden kohdalla lopputuloksessa on varmasti ykkönen.

ORA-käsky on kätevä silloin, kun haluat varmistaa, että tietyssä tavussa tietty bitti on ykkönen, mutta et halua vaikuttaa muihin bitteihin lainkaan.

Seuraava esimerkki varmistaa, että muistissa olevan luvun 5. bitti on ykkönen:

**\$0350 LDA \$FB** (lataa akkuun luku osoitteesta 251 (\$00FB). Huomaa nollasivun osoitusmuoto: osoitteessa vain kaksi heksanumeroa)

**\$0352 ORA #\$10** (suorita TAI-operaatio akun ja luvun 00010000 välillä)

**\$0354 STA \$FB** (vie luku takaisin paikalleen)

**\$0356 RTS** (jos kokeilet esimerkkiä, on sinun syytä lisätä loppuun RTS, joka palauttaa kontrollin takaisin sinne, mistä konekieliohjelmaa kutsuttiin)

Yleensä AND ja ORA -käskyjä käytetään yhdessä manipuloimaan haluttua bittiä. Kun bitti halutaan asettaa (muuttaa ykköseksi), käytetään ORAa ja kun se halutaan nollata, käytetään ANDiä.

Kolmas samansukuinen käsky on EOR. Se on lyhenne sanoista Exclusive OR ja tarkoittaa suomeksi 'ehdoton TAI'. Sekin vertailee kahta lukua bitti bitiltä ja toimii samoin kuin TAI-operaatiokin, eli tuloksessa bitti on ykkönen, jos jompikumpi lähtöbiteistä on ykkönen.

EORin ehdottomuus paljastuu kuitenkin siitä, että se hyväksyy tilanteen vain, jos jompikumpi biteistä on ykkönen, mutta EI MOLEMMAT. Jos sekä akussa että muistista haettavassa luvussa saman paikan bitit ovat ykkösiä, on tuloksessa sillä kohtaa nolla.

EORin toimintaa havainnollistaa parhaiten sen totuustaulukko:

**EOR: 0 ehd.TAI 0 = 0**

**0 ehd.TAI 1 = 1**

**1 ehd.TAI 0 = 1**

**1 ehd.TAI 1 = 0**

**akun sisältö: 00001111**

**vertailuluku: 00110011**

**akkuun jäävä luku: 00111100**

Totuustaulukosta näet, että jos vertailuluvun bitti on nolla, pysyy akku muuttumattomana sen bitin kohdalla. Jos taas vertailuluvussa bitti on ykkönen, 'kääntää' se vastaavan akun bitin arvon: nolla muuttuu ykköseksi ja päinvastoin.

EOria käytetään suorittamaan kytkintoimintoa (engl. toggle). Jos akussa olevaa lukua EORataan useaan kertaan samalla luvulla, muuttuvat ykkösten kohdalla akun bitit vuoroin ykkösiksi ja vuoroin nolliksi.

|                          |                                                                                                                                                                                                             |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>\$0350 LDA \$0370</b> | (haetaan käsiteltävä luku muistipaikasta \$0370)                                                                                                                                                            |
| <b>\$0353 EOR #\$01</b>  | (ehdoton TAI akun ja \$01:n välillä. \$01:ssä ensimmäinen bitti on ykkönen, muut ovat nolliä. Oli akun ensimmäinen bitti ykkönen tai nolla, muuttaa se tilansa toiseksi muiden bittien säilyessä ennallaan) |
| <b>\$0355 EOR #\$01</b>  | (akun ensimmäinen bitti muuttuu nyt alkuperäiseen tilaansa)                                                                                                                                                 |
| <b>\$0357 EOR #\$01</b>  | (akun 1. bitti muuttuu jälleen...)                                                                                                                                                                          |
| <b>\$0359 STA \$2050</b> | (viedään luku takaisin)                                                                                                                                                                                     |
| <b>\$035B RTS</b>        | (palataan takaisin)                                                                                                                                                                                         |

## Testaukset BIT, CMP, CPX, CPY

Yhteenlasku, vähennyslasku ja loogiset käskyt suorittavat kaikki todellisen operaation: akun sisältämää lukua manipuloidaan ja saatu tulos siirretään akun uudeksi sisällöksi.

Testaavat käskyt BIT, CMP, CPX ja CPY suorittavat samanlaisia toimintoja, mutta ne ovat piilo-operaatioita — akun tai muiden rekistereiden sisältöä ei muuteta. Tulos jää ikäänkuin roikkumaan ilmaan ja ainoa näkyvä muutos on tilarekisterin lippujen asettuminen ilmaan jäävää tulosta vastaavaksi. Z-lippu asetuu, jos tulos olisi ollut nolla ja päinvastoin. Samoin käy N ja C -lipuille.

CMP on näistä ehdottomasti yleisin käsky. Se vähentää akun sisällöstä muistista haetun luvun (mutta ei laita tulosta akkuun). Jos akku ja haettu luku ovat yhtä suuria, asetuu Z- eli nollalippu ykköseksi, koska niiden erotus on nolla. CMP-käskyssä käytetään varsin usein välitöntä osoitusmuotoa, eli vertailuluku on tunnettu vakioluku, joka sijaitsee ohjelmassa heti CMP-käskyn perässä.

Vertailukäskyä tarvitaan toistuvasti esimerkiksi peleissä liikuteltaessa hahmoja kuvaruudulla. Kun peliohjainportista olemme havainneet, että alusta halutaan siirrettävän vasemmalle, on meidän luonnollisesti ensin testattava kuvaruudulta, onko vasemmanpuoleinen paikka vapaa vai törmääkö alus johonkin esteeseen.

Vapaat paikat ovat kuvaruudulla yleensä taustan värisiä eli ne vastaavat merkkikoodissa välilyöntiä. ASCII-merkinä (ja myös Commodoren kuvaruutukoodina) välilyönnin koodi on 32 (heksana \$20). Jos siis haemme kuvaruutumuistista vasemmanpuoleisesta merkkipaikasta sen arvon ja vertaamme sitä lukuun 32, voimme päätellä siirrämmekö alusta vai siirrymmekö räjäytyskuvan piirtoon

(törmäyksen seurauksena).

Konekielellä tehtävän voisi suorittaa vaikkapa näin:

**\$033E LDY # \$00** (sijoitetaan Y-rekisteriin nolla)  
**\$0340 LDA (\$FB), Y** (aluksen sijaintia kuvaavaa osoitetta pidämme nollasivulla osoitteissa \$FB ja \$FC, vähemmän merkitsevä tavu \$FB:ssä. Nyt ne sisältävät testattavan paikan osoitteen, mikäli peliohjaimen lukemisen jälkeen olemme jo vähentäneet osoitteesta yhden. (\$FB), Y tarkoittaa epäsuoraa indeksoitua osoitusmuotoa, eli varsinainen osoite, josta haetaan luku akkuun saadaan, kun muistipaikoista \$FB ja \$FC haetaan kaksitavuinen osoite ja siihen lisätään Y-rekisterin sisältö. Y-rekisteri asetettiin nollaksi, koska haluamme juuri osoittimen paikan sisältämän tiedon emmekä mitään seuraavista.)  
**\$0342 CMP # \$20** (verrataan haettua merkkiä \$20:een, joka vastaa välilyöntiä, eli spacea. Akun sisältämä luku ei muutu, mutta li-put asettuvat)  
**\$0344 BEQ \$0349** (mikäli kuvaruudun merkki oli \$20, hypätään muutaman paikan yli ja jatketaan ohjelmaa piirtämällä alus uuteen paikkaan)  
**\$0346 JMP \$0360** (hypätään ohjelmanosaan, joka piirtää aluksen paikalle räjähdyskuvion)  
**\$0349 ... ..** (piirretään alus uuteen paikkaan yhden merkin verran vasemmalle)

CPX ja CPY -käskyjä käytetään harvemmin, vaikkakin ne toimivat täsmälleen samoin kuin CMP. Yleensä niitä käytetään testaamaan indeksirekisterien sisältöä itseään. CPX:llä ja CPY:llä on vain kolme osoitusmuotoa — välitön, absoluuttinen ja nollasivu — kun taas CMP:llä on näiden lisäksi vielä kaikki indeksoidut osoitusmuodot.

Edellinen esimerkki olisi voitu aivan hyvin tehdä toisella tavalla:

\*  
\*  
\*

**\$0340 LDA # \$20** (akkuun välilyönnin koodi, \$20)  
**\$0342 CMP (\$FB), Y** (verrataan akun sisältöä epäsuoran osoitteen avulla kuvaruutumuistin merkkipaikan sisältöön)

\*  
\*  
\*

## Pyöritä, pyöritä... ASL, LSR, ROL, ROR

Viimeinen käsiteltävä käskysarja, joka suorittaa lukujen muokkausta, muodostuu pyörittäjistä. LSR, ASL, ROL ja ROR kaikki siirtävät muistipaikan tai akkurekisterin sisältämiä bittejä aina yhden pykälän verran johonkin suuntaan.

Aivan aluksi tällä ei tunnu olevan paljonkaan merkitystä. Milloin meidän tarvitsi siirrellä bittejä ja mistä syystä? Niillekin löytyy silti käyttöä, LSR ja ASL ovat käteviä esimerkiksi kerto- tai jakolaskussa sekä koodimuunnoksia tehtäessä (seuraavasta kappaleesta löydät esimerkkejä).

ASL, Arithmetic Shift Left eli aritmeettinen siirto vasempaan toimii seuraavasti:

### C-lippu

|       |   |          |                              |
|-------|---|----------|------------------------------|
| ASL   | 1 | 11001100 | luku 204, C-lippu on ykkönen |
| tulos | 1 | 10011000 |                              |

C-lipussa ollut ykkönen hävisi taivaan tuuliin, mutta luvun viimeinen ykkönen siirtyi sen paikalle, uutena bittinä oikealta syötettiin nolla. Luvun arvoksi tuli 152.

Siirron kohteena voi osoitusmuodosta riippuen olla joko akkurekisteri tai jokin muistipaikka. Bittejä siirretään vasemmalle eli enemmän merkitsevään suuntaan ja oikealta syötetään vähiten merkitsevään bittipaikkaan nolla.

Luvun pitäisi siis käytännössä suurentua, koska bittejä siirretään enemmän merkitsevään suuntaan, mutta nyt luku pieneni 204:stä 152:een. Eniten merkitsevä bitti kuitenkin putosi rekisteristä pois ja se siepattiin C-lippuun talteen. Jos siirto suoritetaan ohjelmallisesti voidaan C-lippu testata ja pitää sen arvona 256. Jos se lisätään 152:een, saadaan 408, joka on tasan kaksi kertaa alkuperäinen luku.

LSR, Logic Shift Right eli looginen siirto oikealle toimii samalla tavoin kuin ASL, mutta päinvastaiseen suuntaan. Bittejä siirretään vähemmän merkitsevään suuntaan ja vapautuvalle eniten merkitsevälle bittipaikalle syötetään nolla. Pois putoava bitti otetaan jälleen talteen C-lippuun:

### C-lippu

|       |          |   |                               |
|-------|----------|---|-------------------------------|
| LSR   | 11001100 | 1 | lukua 204 siirretään oikealle |
| tulos | 01100110 | 0 |                               |

C-lipussa ollut ykkönen menetettiin ja tilalle tuli nolla. Luvun arvo on nyt 102.

LSR jakaa luvun kahdella, kuten esimerkistä huomataan:  $204/2 = 102$ . Jos C-lippuun olisi siirtynyt ykkönen ei tulos olisi välittömästi ollut oikein vaan meidän olisi ohjelmallisesti ollut huomioitava se "puolikkaana bittinä" aivan samoin kuin vasemmalle siirrettäessä 256:ena.

Edellisissä siirtokäskyissä yksi bitti (C:n sisältö) katosi kokonaan ja uutena bittinä syötettiin sisään nolla. ROR ja ROL toimivat toisella tavalla, ne eivät puto-



ta pois yhtään bittiä, vaan C-lipussa olevaa bittiä käytetään syöttämään tavun toisesta päästä vapautuva paikka:

**C-lippu**

|              |          |                 |                                   |
|--------------|----------|-----------------|-----------------------------------|
| <b>ROL</b>   | <b>1</b> | <b>11001100</b> | <b>rullataan tavua vasemmalle</b> |
| <b>tulos</b> | <b>1</b> | <b>10011001</b> |                                   |

C-lipussa ollut ykkönen siirtyi nyt vähiten merkitsevän bitin paikalle ja eniten merkitsevä siirrettiin C-lippuun.

**C-lippu**

|              |          |                 |                         |
|--------------|----------|-----------------|-------------------------|
| <b>ROR</b>   | <b>1</b> | <b>11001100</b> | <b>rullaus oikealle</b> |
| <b>tulos</b> | <b>0</b> | <b>11100110</b> |                         |

Vähiten merkitsevä bitti siirtyi C-lippuun.

Rullauskäsky ROR ja ROL eivät tuhoa tietoa ja jos siirto suoritetaan 9 kertaa peräjälkeen, on tuloksena sama luku kuin ennen siirtoja. ASL ja LSR sen sijaan tuhoavat yhden bitin kerrallaan (tosin talteen Carryyn) ja jos siirto suoritetaan 8 kertaa, on tuloksena rekisterin nollaantuminen.

## INCrement ja DECrement

Inkrementointi tarkoittaa kasvattamista ja dekrementointi vähentämistä. Ne ovat käskyjä, joilla voi lisätä muistipaikan sisältöön yhden tai vähentää siitä yhden.

Sama tehtävä voitaisiin suorittaa myös lataamalla muistitavu akkuun ja vähentämällä siitä SBC-käskyllä tai lisäämällä ADC-käskyllä yhden, mutta INC ja DEC ovat lyhyempiä tapoja tehdä sama asia. Lisäksi meidän ei tarvitse tallettaa akussa mahdollisesti olevaa tärkeää lukua tämän toimituksen vuoksi.

Eniten käytetyt inkrementointi- ja dekrementointikäskyt ovat kuitenkin suoraan indeksirekistereihin kohdistuvat INX, INY, DEX ja DEY. Ne ovat vain yhden tavun pituisia ja todella tehokkaita käskyjä luojuja eli silmukoita ohjelmoitaessa. Näitä käskyjä on käytetty tämän kirjan esimerkeissä runsaasti.

Konekielessä X-rekisteriä käytetään usein laskurina, jos meidän on suoritettava jokin ohjelman osa useaan kertaan. Se vastaa aivan Basicin FOR...NEXT silmukassa olevaa laskuria. Basicin lausetta

**FOR X=1 TO 5: NEXT X**

vastaa konekielessä rakenne:

|               |                   |                                                                                                         |
|---------------|-------------------|---------------------------------------------------------------------------------------------------------|
| <b>\$0340</b> | <b>LDX #05</b>    | (ladataan X-rekisteriin kierrosluku, 5)                                                                 |
| <b>\$0342</b> | <b>...</b>        | (tässä on ohjelmanpätkä, joka toistetaan 5 kertaa)                                                      |
| <b>\$034A</b> | <b>DEX</b>        | (vähennetään kierroslaskuria yhdellä)                                                                   |
| <b>\$034B</b> | <b>BNE \$0342</b> | (mikäli kierroslaskuri ei vielä yltänyt nollaan, hypätään suorittamaan edellinen koodi vielä uudestaan) |

Basicissa muuttujan X arvoa liu'utettiin ykkösestä viitoseen, konekielessä

päinvastoin. Tämä johtuu siitä, että hyppykäskyillä (BNE, BEQ) testataan rekisterin muuttumista nolaksi. Luonnollinen liu'utussuunta on siis alaspäin.

Laskuri voidaan toki konekielessäkin pistää pyörimään ylöspäin, esimerkiksi seuraavasti:

|        |            |                                             |
|--------|------------|---------------------------------------------|
| \$0340 | LDX #\$00  | (ladataan laskurin alkuarvoksi nolla        |
| \$0342 | ... ..     | (tähän toistettava lohko)                   |
| \$034A | INX        | (kasvatetaan laskuria yhdellä)              |
| \$034B | CPX #\$05  | (joko tuli vitoseksi?)                      |
| \$034D | BNE \$0342 | (jos ei, hypätään toistamaan ohjelmalohkoa) |

Tässä tapauksessa konekielinen ohjelma pitenee yhdellä vertailukäskyllä (ja samalla hiukan hidastuu).

Kirjoitettaessa tekstiä kuvaruudulle on kuitenkin selvintä käyttää jälkimmäistä toistorakennetta. Jos tiedämme kuvaruutumuistin osoitteen (kursorin paikan), josta tekstimme alkaisi, voimme indeksoidulla osoitusmuodolla (esim. absoluuttinen,X) siirtää tekstin merkki kerrallaan halutulle paikalleen.

Jos olemme sijoittaneet tekstimme muistiin muistipaikasta \$2C00 lähtien ja tiedämme, että se on 12 merkkiä pitkä, voimme siirtää tekstin kuvaruudulle seuraavalla rutiinilla. Sijoitamme tekstin alkamaan kuvaruudun ensimmäisestä merkkipaikasta. Commodore 64:ssä kuvaruutumuisti alkaa osoitteesta \$0400:

|        |              |                                                                                                                                                                                      |
|--------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$0340 | LDX #\$00    | (nollataan X-rekisteri)                                                                                                                                                              |
| \$0342 | LDA \$2C00,X | (ladataan ensimmäinen merkki ensimmäisestä muistiosoitteesta)                                                                                                                        |
| \$0345 | STA \$0400,X | (sijoitetaan kuvaruudulle)                                                                                                                                                           |
| \$0348 | INX          | (kasvatetaan X-rekisteriä eli siirrytään seuraavaan merkkipaikkaan niin tekstin lähtöalueella kuin kuvaruudullakin. X-rekisteri hoitaa merkki merkiltä etenemisen molemmissa päissä) |
| \$0349 | CPX #\$0C    | (oliko 12. merkki?)                                                                                                                                                                  |
| \$034A | BNE \$0342   | (jos kaikkia merkkejä ei vielä siirretty, palataan osoitteeseen \$0342)                                                                                                              |
| \$034C | ...          | (jatketaan ohjelmaa eteenpäin)                                                                                                                                                       |

## HAARAUTUMISET JA HYPYT:

**JMP, JSR, RTS, RTI,  
BNE, BEQ, BCC, BCS**

Basicissa haarautumiset tehdään GOTO ja GOSUB-käskyillä. GOTO haarautuu pysyvästi uuteen osoitteeseen, GOSUB vain käväisee suorittamassa aliohjelman,

josta palataan takaisin RETURN -käskyllä.

Konekielessä on aivan vastaavat käskyt, JMP ja JSR. JMP hyppää annettuun osoitteeseen ja jatkaa ohjelman suoritusta sieltä. JMP-käskyä seuraa kaksitavui-nen osoite, jossa jälleen vähemmän merkitsevä tavu seuraa ohjelmakoodissa en-sin. Käytännön työskentelyssä annetaan assemblerille osoite oikeassa muodossa, esimerkiksi JMP \$0340, ja assembler-ohjelma huolehtii osoitteen tavujen kään-tämisestä prosessorin ymmärtämään järjestykseen.

JMP-käskyllä on tämän absoluuttisen osoitusmuodon lisäksi vielä eräs erittäin harvoin käytetty osoitusmuoto, jota ei itse asiassa ole millään muulla käskyllä, nimittäin epäsuora osoitusmuoto ilman minkäänlaista indeksointia. Lisäksi JMP:n epäsuora osoitusmuoto voi hakea lopullisen osoitteen mistä kohtaa muis-tia tahansa. Indeksoidut epäsuorat osoitusmuodotahan kykenivät hakemaan lo-pullisen osoitteen ainoastaan nollasivun alueelta.

JMP (\$FE00) sisältää siis kahden tavun mittaisen alkuperäisen osoitteen. Lo-pullisen osoitteen vähemmän merkitsevä tavu haetaan osoitteesta \$FE00 ja enemmän merkitsevä tavu osoitteesta \$FE01.

Tähän osoitusmuotoon liittyy myös 6502:n pieni bugi eli virhetoiminta: alku-peräisen osoitteen vähemmän merkitsevä tavu ei saa olla 255 eli FF tai prosessori ei toimi oikein. Tämän yhdistelmän välttäminen on oleellista, mutta toisaalta konekielissä ohjelmissa tuskin koskaan käytetään hyppykäskyn epäsuoraa osoi-tusmuotoa, joten siitä ei tarvitse sen kummemmin välittää.

JSR-käsky (Jump to SubRoutine, hyppää aliohjelmaan) hyppäyttää prosesso-rin suorittamaan aliohjelman, josta palataan, kun ohjelmassa esiintyy RTS (Re-Turn from Subroutine) -käsky. Aliohjelmaan lähdettäessä prosessori tallentaa pinorekisteriin sen hetkisen ohjelmalaskurin arvon ja suorittaessaan RTS-käskyä prosessori palauttaa ohjelmalaskuriin tämän arvon pinosta. Tämän vuoksi on tärkeää pitää huolta siitä, että aliohjelmassa ei pinosta poisteta ylimää räisiä lu-kuja tai sinne ei jätetä tarpeetonta "roskaa", joka sekoittaa ohjelman kulun oh-jelmalaskurin arvoa palautettaessa.

## Ohjelmoinnin perusta: BNE, BEQ, BCC, BCS

Otsikossa mainittuja ehdollisia hyppykäskyjä voi hyvällä syyllä sanoa ohjel-moinnin perustaksi. Ilman niitä ohjelmamme eivät voisi tehdä mitään kovin fik-sua. Ne mahdollistavat ohjelman haarautumisen eri tilanteissa eri tavalla ja saa-vat ohjelmat joskus tuntumaan "älykkäiltä" koska ne tuntuvat tekevän älykkäitä päätelmiä.

Ehdollisten hyppykäskyjen toiminta perustuu keskusyksikön muistina toimi-vaan tilarekisteriin, joka pystyy kertomaan jotain menneistä tapahtumista. Tila-rekisterihän pystyi osoittamaan, oliko edellisen toimenpiteen tuloksena rekisterin nollautuminen vai ei, asettiko edellinen toimenpide muistinumeron (Carry) vai ei, oliko edellinen tulos negatiivinen vai ei jne.

Useat käskyt eivät vaikuta tilarekisteriin, joten muisti ulottuu usein pidemmällekin kuin juuri edelliseen toimenpiteeseen.

BEQ (Branch if EQual) ja BNE (Branch if Not Equal) testaavat ensin vastaavasti onko nollalippu (Z) asettunut vai ei ja suorittavat haarautumisen, mikäli ehto oli paikkansa pitävä. Kun lippu on ykkönen, sen sanotaan olevan 'asettunut'.

BEQ ja BNE ovat yleisimmät haarautumiskäskyt ja niitä käytetään tiuhaan FOR...NEXTiä vastaavissa konekielen toistorakenteissa. Kun laskurina toimivaa indeksirekisteriä vähennetään yhdellä ja se tulee nollassa, asettuu nollalippu ykköseksi ja se voidaan BEQ:lla testata.

Usein BEQ:ta käytetään myös testaamaan, oliko akkuun ladattu luku sattumalta nolla, liput kun asettuvat myös latauskäskyjen suorituksessa (LDA, LDX, LDY).

Vaikka BNE ja BEQ -käskyjä käytetäänkin konekielessä usein samassa yhteydessä kuin FOR...NEXTiä Basicissa, niin niiden suora vastine Basicissa on kuitenkin IF...THEN -käsky. Esimerkiksi konekielen käskyriivi

**\$0346 BEQ \$0360    (jos Z-lippu = 1, hyppää osoitteeseen \$0360)**

vastaa täysin Basic-riviä

**1700 IF A=1 THEN 1900    (jos A=1, hyppää riville 1900).**

Toinen ehdollisten hyppykäskyjen ryhmä testaa muistinumeroa eli Carry-lippua. BCC haarauttaa ohjelman, mikäli C-lippu on nolla ja BCS vastaavasti, jos se on ykkönen.

Yleensä C-lippu asettuu yhteenlaskun yhteydessä, kun tulos on liian suuri mahtuakseen kokonaan akkuun. Tuloksen ylimääräinen bitti johdetaan silloin C-lippuun, jotta ohjelma voi ottaa sen huomioon. Vähennyslaskussa taas C-lipun nollautuminen on merkinä "lainanumeron" etenemisestä yli akun reunan.

Ehdollisilla hyppykäskyillä voidaan testata vielä kahta muuta tilarekisterin lippua, N (Negative = negatiivinen) ja V (oVerflow = ylivuoto). Ne liittyvät kuitenkin etumerkillisten lukujen (positiiviset ja negatiiviset luvut) aritmetiikkaan ja ne voidaan tavallisissa konekielisissä ohjelmissa unohtaa.

N-lippu asettuu, mikäli toimenpiteen tuloksena rekisterin tai muistipaikan eniten merkitsevä bitti oli ykkönen. Etumerkillisessä aritmetiikassa tämä bitti (bitti 7) on varattu etumerkille eikä siis vaikuta itse luvun suuruuteen. Laskettaessa yhteen etumerkillisiä lukuja ei C-lipulla ole enää samaa merkitystä kuin normaalisti, koska siihen siirtyy myös etumerkkien aiheuttama muistinumero. Sen vuoksi tilarekisterissä on etumerkillisten lukujen muistinumero, V eli ylivuoto, joka asettuu, mikäli toiseksi eniten merkitsevistä biteistä (bitti 6) syntyy muistinumero.

Joka tapauksessa näitä lippuja testaavat käskyt BMI (haaraudu, jos bitti 7 on

ykkönen), BPL (haaraudu, jos bitti 7 on nolla), BVC (haaraudu, jos ei 6. bitin ylivuotoa) ja BVS (haaraudu, jos 6. bitin ylivuoto) voidaan unohtaa ja keskittyä toimimaan nolla- ja carrylippujen turvin.

Eräs paha houkutus näiden käskyjen käytölle tulee kuitenkin usein esiin ja sitä on syytä välttää. Jos nimittäin testaamme akkua käskyllä CMP #\$20 ja haluamme tietää onko akussa oleva luku pienempi vai suurempi kuin tuo \$20, tulee mieleen varsin helposti, että se voidaan testata käskyillä BMI ja BPL.

CMP-käskyhän suoritti akkuun kohdistuvan vähennyslaskun ja jos \$20 oli suurempi, voisi ajatella tuloksen menneen negatiiviseksi ja päinvastoin.

Näin ei kuitenkaan ole asia eikä ole syytä sekoittaa etumerkillisen aritmetiikan käskyjä normaaleihin laskutoimituksiin.

Oletetaan esimerkiksi, että akussa olisi ollut luku 214. Vähennetään siitä tuo 32 (\$20), jolloin saamme 182. Tulos ei ollut lähelläkään mennä negatiivisen puolelle, mutta luku 182 on bitteinä 10110110. Eniten merkitsevä bitti on ykkönen, joten jos käytämme BMI-käskyä, tulkitsee keskusyksikkö tuloksen siitä huolimatta negatiiviseksi! Varo siis näitä käskyjä, ellet todella käsittele etumerkillisiä lukuja.

Jos haluat testata "vähemmän kuin" -tilannetta, käytä haarautumiskäskyä BCC. Vastaavasti "enemmän kuin" -tilannetta testataan BCS:llä.

Seuraava ohjelmanpätkä näyttää, kuinka testejä voi olla peräkkäinkin kaikkien tilanteiden läpikäymiseksi:

|        |     |        |                                                      |
|--------|-----|--------|------------------------------------------------------|
| \$0341 | LDA | \$0700 | (ladataan akkuun testattava luku osoitteesta \$0700) |
| \$0344 | CMP | #\$20  | (verrataan akkua 32:een)                             |
| \$0346 | BEQ | \$0360 | (akku=32, yhtäsuuruustilanteessa hypätään \$0360)    |
| \$0348 | BCC | \$0370 | (akku < 32, pienempi kuin -tilanteessa hyppy \$0370) |
| \$034A | BCS | \$0380 | (akku > 32, suurempi kuin tilanteessa hyppy \$0380)  |

## Keskeytykset

Keskusyksikön käskykannassa on muutama käsky (RTI ja BRK), jotka liittyvät ns. keskeytysten käsittelyyn.

RTI on käsky, jolla keskeytysohjelmasta palataan takaisin siihen kohtaan, missä keskeytys tapahtui. BRK on taas käsky, jolla ohjelmallisesti saadaan aikaan keskeytys.

Mitä sitten keskeytykset ovat? Muistat kirjan neljännestä luvusta maininnat parista keskusyksikköön tulevasta linjasta, "johdosta", joita kutsutaan nimillä IRQ ja NMI. Ne ovat ulkoisille laitteille tai elektronisille piireille tarkoitettuja

viestilinjoja, joilla nämä voivat kertoa keskusyksikölle, että niillä on jotakin todella tärkeää tapahtumassa — jotain sellaista, joka vaatii erityistoimenpiteitä heti eikä viidestoista päivä.

Normaalitilassa IRQ ja NMI -linjat ovat ykkösiä (jännitteellisiä). Jos jokin ulkoinen laite tai keskusyksikköä ympäröivistä apupiireistä syöttää IRQ- tai NMI-linjaan nollan (ei jännitettä, eli 'vetää maahan'), keskusyksikkö huomaa sen ja toimii seuraavasti.

Oli keskusyksikkö sitten tekemässä mitä tahansa, se keskeyttää tämän toiminnan. Juuri suorituksessa oleva käsky suoritetaan tosin loppuun asti, mutta sen jälkeen keskusyksikkö tallentaa ohjelmalaskurin ja tilarekisterin sisällön pino-muistiin. Tämän jälkeen keskusyksikkö alkaa suorittaa keskeytysohjelmaa. Tämän keskeytysohjelman alkuosoitteen keskusyksikkö löytää tietyiltä vakiopaikoilta muistialueen loppupäässä. IRQ:lla ja NMI:llä on kummallakin oma osoitepaikkansa,

**\$FFFFE ja \$FFFF (IRQ)**  
**\$FFFFA ja \$FFFFB (NMI).**

Keskusyksikkö voi siis välillä keskeyttää ohjelmasi läpikäymisen, siirtyä suorittamaan keskeytysohjelmaa, ja sitten palata takaisin ohjelmasi pariin ikään kuin mitään ei olisi tapahtunutkaan.

Itse asiassa tietokoneessasi tapahtuu näin kaiken aikaa. Tietokoneesi oma käyttöjärjestelmä nimittäin käyttää tehokkaasti hyväkseen näitä keskeytystoimintoja. Keskeytyksiä tapahtuu kymmeniä kertoja sekunnissa mm. kellon pitämiseksi ajassa. Keskeytyksen aikana kelloa kuvaavia muistipaikkoja käydään kasvattamassa, suoritetaan muutamia muitakin esimerkiksi näppäimistön lukuun liittyviä toimintoja, ja palataan sitten taas "pääohjelmaan".

Keskeytyksistä ei ole sinulle mitään haittaa edes konekielellä ohjelmoidessasi. Ne ovat niin lyhyitä, etteivät ne vaikuta ohjelman näennäiseen toimintaan lainkaan.

Konekielellä ohjelmoidessasi voit sen sijaan käyttää itekin keskeytyksiä hyväksesi. Voit esimerkiksi lisätä käyttöjärjestelmän suorittamaan keskeytysohjelmaan oman pienen rutiinisi, joka suoritetaan aina keskeytysrutiinin yhteydessä. Oma keskeytysrutiinisi voi esimerkiksi käydä ohjaamassa äänigeneraattoria, jolloin saat muihin ohjelmiisi jatkuvaa taustaaääntä riippumatta siitä, mitä ohjelmasi tekee ja missä vaiheessa se on.

Keskeytyksien hyväksikäyttö edellyttää kuitenkin hyvää ohjelmointitaitoa ja oman laitteensa tuntemusta.

Keskeytyksiin liittyy muutamia huomioitavia asioita. NMI (Non Maskable Interrupt, ei-estettävä keskeytys) on keskeytys, jonka suoritusta et voi millään konekielen tempulla estää. Se käynnistyy aina kun NMI-linja muuttuu nollassa. IRQ (Interrupt ReQuest, keskeytyspyyntö) voidaan sen sijaan 'maskata' eli sen suori-

tus voidaan estää.

Tilarekisterissä on eräs bitti, I-lippu, joka on normaalitilassa 0. Kun tämä bitti muutetaan ykköseksi, ei keskusyksikkö lähdekään suorittamaan keskeytysohjelmaa, vaan on kuin ei huomaisikaan IRQ-linjalle tuotua keskeytyspyyntöä (nollasignaalia). Kun I-lippuun jälleen viedään nolla, suorittaa keskusyksikkö keskeytysohjelman pyynnön tullessa. Ohi mennyttä pyyntöä se ei kuitenkaan muista. Jos keskeytys on ollut kielletty pyynnön tullessa ja IRQ-linja nousee takaisin ykköseksi ennenkuin keskeytykset jälleen sallitaan, ei keskeytysrutiini tule suoritetuksi.

RTI-käskyyn törmäät, jos tutkit laitteesi käyttöjärjestelmää disassemblerilla. Itse et sitä ohjelmoidessasi kuitenkaan tarvitse. Vaikka tekisit itse oman lisäyksesi keskeytysohjelmaan, on sinun syytä aina oman lisäyksesi viimeiseksi käskyksi laittaa hyppykäsky käyttöjärjestelmän sisältämään keskeytysohjelmaan, jotta se saisi omat tehtävänsä suoritettua. Siellä ohjelmassa tulee ennen pitkää vastaan RTI, joka palauttaa keskusyksikön keskeytyneeseen pääohjelmaan.

Jos kuitenkin teet kokonaan oman keskeytysohjelman ja haluat, ettei käyttöjärjestelmän keskeytysohjelmaa suoriteta lainkaan, on keskeytysohjelmasi viimeisen käskyn oltava RTI. Keskeytyksen sattuessahan keskusyksikkö tallensi pinoon sen hetkiset arvot ohjelmalaskurista ja tilarekisteristä. Kun keskeytysohjelmasta palataan, on nämä tietenkin palautettava oikeille paikoilleen, jotta pääohjelman suoritus voisi jatkua ilman hämminkiä. RTI-käsky tekee juuri tämän.

Rekisterien tallennukseen liittyy vielä eräs keskeytyksissä huomioitava asia: keskusyksikkö ei tallenna akun eikä X ja Y -rekisterien arvoja lähtiessään suorittamaan keskeytysohjelmaa. Se on tehtävä omassa keskeytysohjelmassa, mikäli se muuttaa näiden rekisterien arvoja, sillä palattaessa keskeytyksestä takaisin pääohjelmaan eivät asiat enää olisikaan samalla mallilla kuin lähtiessä.

Siis ensiksi keskeytysohjelmassa: tallenna A, X ja Y, viimeiseksi keskeytysohjelmassa: palauta Y:n, X:n ja A:n arvot paikoilleen. Sen voi tehdä esimerkiksi pinon akkurekisterin kautta:

|                   |                                                                                                     |
|-------------------|-----------------------------------------------------------------------------------------------------|
| <b>\$2000 PHA</b> | (tallenna akku pinoon)                                                                              |
| <b>\$2001 TXA</b> | (siirrä X-rekisteri akkuun)                                                                         |
| <b>\$2002 PHA</b> | (vie akku pinoon)                                                                                   |
| <b>\$2003 TYA</b> | (siirrä Y-rekisteri akkuun)                                                                         |
| <b>\$2004 PHA</b> | (vie akku pinoon)                                                                                   |
| <b>*</b>          |                                                                                                     |
| <b>*</b>          |                                                                                                     |
| <b>*</b>          |                                                                                                     |
| <b>\$2500 PLA</b> | (ohjelman päättyessä rekisterit palautetaan akun kautta. Ensin luku pinosta akkuun)                 |
| <b>\$2501 TAY</b> | (ja akku Y-rekisteriin. Huomaa käänteinen järjestys: viimeiseksi pinoon viety haetaan sieltä ensin) |



|                   |                                                                            |
|-------------------|----------------------------------------------------------------------------|
| <b>\$2502 PLA</b> | (seuraava luku pinosta akkuun)                                             |
| <b>\$2503 TAX</b> | (ja akku X-rekisteriin)                                                    |
| <b>\$2504 PLA</b> | (palauta akun sisältö)                                                     |
| <b>\$2504 RTI</b> | (jos teet kokonaan oman keskeytysohjelmasi, on sen päätyttävä RTI-käskyyn) |

Vaikka keskusyksikkö ei suorita rekisterien tallennuksia keskeytyksen sattuessa, Commodore 64:n käyttöjärjestelmä tekee sen automaattisesti ennen varsinaisen keskeytysohjelman käynnistämistä. Jos siis teet vain oman lisäyksesi keskeytysrutiiniin alkuun, ei sinun tarvitse huolehtia rekisterien tallennuksesta.

BRK (BReaK, keskeytyks) on käsky, jolla voit ohjelmassasi aiheuttaa keskeytyksen. BRK:n koodi on \$00 eli binaarisena 00000000. BRK vastaa IRQ-keskeytystä, mutta kun keskusyksikkö törmää siihen ohjelmassa, se asettaa tilarekisterissä olevan B-lipun ykköseksi. Sen ansiosta keskeytysohjelmassa voidaan B-lippua tutkimalla selvittää, oliko kyseessä ohjelmallinen BRK-keskeytys vai tavallinen IRQ-linjalta tuleva keskeytys.

BRK ja tavallinen IRQ-keskeytys asettavat myös I-lipun ykköseksi eli keskeytyksen aikana muut samanlaiset keskeytykset ovat kiellettyjä. I-lipun voi myös nollata keskeytyksen aikana, jolloin päällekkäiset keskeytykset ovat mahdollisia, mutta se vaatii tarkkaa harkintaa, ettei uusi keskeytys tietämättömänä tuhoaisi sitä, mitä ensimmäinen on siihen mennessä saanut aikaan.

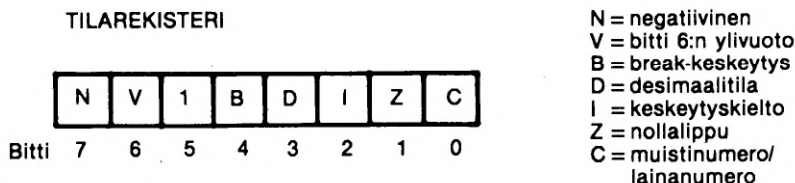
BRK-keskeytyksellä ei näytä olevan mitään merkitystä: samaan tulokseenhan päästäisiin aivan tavallisella aliohjelmakutsulla. BRK on kuitenkin erittäin käytökelpoinen virheiden etsinnässä. Monet konekieliohjelmoinnin apuneuvot, monitoriohjelmat, käyttävät BRK-käskyä keskeyttämään virheelliseksi todettu ohjelma halutussa kohdassa, jotta voidaan ohjelman kulkua seuraamalla jäljittää virheen paikka. Näissä monitoriohjelmissa keskeytysrutiiniksi sijoitetaan ohjelmanpätkä, joka tulostaa kuvaruudulle keskusyksikön rekisterien sisällöt.

Jos (kun) tekemäsi ohjelma osoittautuu virheelliseksi (kone sekoaa tms.), voit tällaista monitoria käyttäessäsi sijoittaa johonkin ohjelmakohtaan siinä olevan todellisen käskyn sijasta BRK:n. Jos ohjelma uudelleen ajattaessa toimii oikein, ja keskusyksikön osuessa BRK:hon keskeytysrutiini tulostaa rekisterit ruudulle sellaisina kuin niiden pitäisikin olla, voit olla varma, että virhe löytyy tämän BRK-kohdan jälkeen. Muutteleamalla BRK:n paikkaa voit paikallistaa virheen sijainnin tarkasti.

Keskeytysten hyväksikäyttö kuuluu kuitenkin edistyneempään konekieliohjelmointiin ja on parasta pidättäytyä siitä, kunnes on tullut tutummaksi tavallisen 'suoran' konekielen kanssa.

## LIPUT LIEHUMAAN: CLC, CLD, CLI, CLV, SEC, SED ja SEI

Tilarekisteriä voidaan manipuloida yksittäisillä käskyillä. Tilarekisterihän toimii keskusyksikön 'muistina', joka tosin ei ulotu kovin pitkälle — korkeintaan muutaman käskyn päähän tavallisissa ohjelmissa.



*Tilarekisterin lippuja käytetään ohjelmassa hyväksi haarautumiskäskyjen avulla. Tilarekisterin liput "muistavat" aiempia tapahtumia, oliko luku nolla vai ei, oliko keskeytykset kielletty jne. Tilarekisterin bitti 5 on aina ykköstilassa.*

C-lippu pitää sisällään tietoa siitä, tuliko edellisestä laskutoimituksesta muistinumeroa vai ei. Yhteenlaskussa (ADC) muistinumeroa vastaa ykkönen C-lipussa, vähennyslaskussa (SBC) muistinumeroa eli 'lainanumeroa' vastaa C-lipun nollatila laskutoimituksen jälkeen.

Ennen yhteenlaskua C-lippu on kuitenkin nollattava, jotta siihen vahingossa jäänyt ykkönen ei tulisi yhteenlaskuun mukaan. Se tapahtuu CLC (CLear Carry; nollaa muistinumero) -käskyllä. Jos useita tavuja lasketaan peräkkäin yhteen siten, että tavut muodostavat yhdessä suuremman luvun, ei muistinumeroa tietenkään nollata yhteenlaskujen välillä.

Ennen vähennyslaskua muistinumero asetetaan ykköseksi SEC (SEt Carry, ase-ta muistinumero) -käskyllä. Vastaavasti voidaan heilutella D- ja I-lippuja. I (Interrupt, keskeytys) -lipun ollessa nolla ovat keskeytykset sallittuja ja sen ollessa ykkönen on keskeytykset kielletty.

D-lippu asetetaan, jos laskutoimitusten halutaan tapahtuvan ns. BCD-koodatussa muodossa. BCD (Binary Coded Decimal, binaarisesti koodattu kymmenjärjestelmä) -järjestelmässä 8-bittistä tavua käsitellään kahtena neljän bitin ryhmänä samalla tavalla kuin se voidaan jakaa kahdeksi neljän bitin heksaluvuiksi.

BCD-järjestelmässä tämän neljän bitin muodostaman luvun arvo ei kuitenkaan voi ylittää 9:ää, vaan seuraavalle neljän bitin ryhmälle siirtyy muistinumero jo kymmenen kohdalla, aivan kuin kymmenjärjestelmässäkin. Seuraavassa tau-

lukossa on rinnakkain lukuja BCD:n mukaan ja tavallisina heksoina tulkittavissa olevina binaarilukuina:

| 10-järj. | BCD       | Binaariluku |
|----------|-----------|-------------|
| 1        | 0000 0001 | 0000 0001   |
| 2        | 0000 0010 | 0000 0010   |
| 3        | 0000 0011 | 0000 0011   |
| 4        | 0000 0100 | 0000 0100   |
| 5        | 0000 0101 | 0000 0101   |
| 6        | 0000 0110 | 0000 0110   |
| 7        | 0000 0111 | 0000 0111   |
| 8        | 0000 1000 | 0000 1000   |
| 9        | 0000 1001 | 0000 1001   |
| 10       | 0001 0000 | 0000 1010   |
| 11       | 0001 0001 | 0000 1011   |
| 12       | 0001 0010 | 0000 1100   |
| *        |           |             |
| *        |           |             |
| *        |           |             |
| 27       | 0010 0111 | 0001 1011   |

Jos yhteenlaskut suoritetaan D-lippu ykkösenä eli BCD-tilassa ja akkuun lisätään aina yksi, niin akun sisältämä luku etenee 00,01,02... aina 99:ään asti ja sadan kohdalla akku nollautuu ja C-lippuun siirtyy muistinumero.

Kumpikin akun neljän bitin ryhmästä pitää siis sisällään kymmenjärjestelmän luvun ja laskutoimitukset tapahtuvat kymmenjärjestelmän sääntöjen mukaan.

Tämä ominaisuus on joskus hyödyllinen, esimerkiksi jos kuvaruudulla on pidettävä pistelaskuria, joka voi edetä aina 999999:ään asti. Tämä vaatisi binaari-muodossa useamman tavun, kolme kappaletta. Näin pitkän luvun muuttaminen kuvaruututulostusta varten kymmenjärjestelmään veisi aika tavalla aikaa ja saat-taisi hidastaa ohjelmaa liikaa. Samaan kolmeen tavuun mahtuu myös vastaava luku BCD-muodossa: vähiten merkitsevään tavuun ykköset ja kymmenet, seuraa-vaan sadat ja tuhannet ja eniten merkitsevään tavuun kymmenet tuhannet ja sadat tuhannet:

|                                              |        |        |                                              |   |   |                                              |   |   |
|----------------------------------------------|--------|--------|----------------------------------------------|---|---|----------------------------------------------|---|---|
| 1.tavu                                       | 2.tavu | 3.tavu |                                              |   |   |                                              |   |   |
| <table><tr><td>9</td><td>9</td></tr></table> | 9      | 9      | <table><tr><td>9</td><td>9</td></tr></table> | 9 | 9 | <table><tr><td>9</td><td>9</td></tr></table> | 9 | 9 |
| 9                                            | 9      |        |                                              |   |   |                                              |   |   |
| 9                                            | 9      |        |                                              |   |   |                                              |   |   |
| 9                                            | 9      |        |                                              |   |   |                                              |   |   |

Laskutoimitukset voidaan kaikki suorittaa useamman tavun peräkkäisinä laskuina BCD-muodossa ja tulostettaessa tarvitsee muuttaa vain kukin neljän bitin ryhmä kerrallaan sen sisältämää lukua vastaavaksi kuvaruutukoodiksi.

BCD-laskentaa voidaan parhaiten havainnollistaa esimerkin avulla. Seuraavas-  
sa esimerkissä on sama yhteenlasku suoritettu kaksi kertaa, toinen D-lipun olles-  
sa ykkönen ja toinen D-lipun ollessa nolla. Huomaa, miten luvut tulkitaan eri

tavalla!

| D = 1 |           | D = 0 |           |
|-------|-----------|-------|-----------|
| 18    | 0001 1000 | \$18  | 0001 1000 |
| +23   | 0010 0011 | +\$23 | 0010 0011 |
| <hr/> |           | <hr/> |           |
| 41    | 0100 0001 | \$3B  | 0011 1011 |

## Tyhjäntoimittaja

Vielä on käsittelemättä 56. 6502:n käsky. Koodiltaan se on \$EA ja mnemoniselta nimeltään varsin kuvaava NOP (No OPeration, ei toimintaa). NOP ei todella tee mitään. Kun keskusyksikkö törmää NOPpiin, se kohauttaa olkapäitään ja siirtyy seuraavaan käskyyn.

Silti, BRK:n tapaan, NOPilla on omat tehtävänsä ohjelman rakentamisessa ja debuggauksessa eli vianetsinnässä.

Ohjelmaa rakentaessasi saatat joskus joutua tilanteeseen, jossa sinulla on ohjelmassa parin tavun mittainen aukko keskellä, mutta ei mitään käskyjä sinne pantavaksi. Joko et vielä tiedä, mitä siihen tulee, tai sitten olet vain onnistunut laskemaan tilantarpeesi väärin, etkä halua enää lukuisien hyppykäskyjen takia siirtää ohjelman koko häntäpäätä taaksepäin. Silloin voit täyttää tyhjät kohdat NOP-käskyillä ja ohjelma rullaa aukon yli nikottelematta.

Joskus saattaa pitkissä ja kehittyvissä ohjelmissa olla aivan hyvä ohjelmointitapa jättää aukkoja kohtiin, joissa myöhemmin saattaa tarvita lisätavuja. Näiden tavujen käyttöönotto ei myöhemmin vaikuta ohjelman hyppykäskyihin ja esimerkiksi julkaistut tiedot ohjelmasta pitävät muutoksen jälkeenkin paikkansa. Joidenkin tietokoneiden käyttöjärjestelmiin on tahallaan jätetty juuri tästä syystä NOP-aukkoja.

NOPpia voi käyttää myös virheen etsinnässä. Joissakin tapauksissa vika saattaa löytyä korvaamalla vialliseksi epäilty käsky NOPilla. Kun ohjelma ei tee mitään, ei se myöskään tee virhettä, voisi ajatella. Hyvin usein kuitenkin käskyn peittäminen NOPilla aiheuttaa sekin ohjelmaan virhetoiminnon, joten tilanne on harkittava tarkoin ettei jouduta ojasta allikkoon.

Samoin on huomioitava peitettävän käskyn pituus: käskykoodin lisäksi on peitettävä sitä seuraavat samaan käskyyn kuuluvat osoitustavut eli argumentti NOPilla.

TÄSSÄ ne olivat, kaikki 56 käskyä. Monet niistä ovat harvinaisia ja jotkut suoraan vältettäviä. Kaiken kaikkiaan konekielisessä ohjelmoinnissa käytetään toistuvasti vain vajaata pariakymmentä näistä käskyistä. Osoitusmuodoistakin käytetään pääasiassa vain muutamia yleisempiä.

Nyt tunnet käskysanat, joilla ohjelmat rakennetaan, esimerkeissä on esiintynyt lyhyitä kokonaisia ohjelmiakin. Seuraavassa luvussa kerrotaan, miten näistä palasista kootaan isompia ohjelmia.

## Konekielisen ohjelman rakentaminen

# 7



## Isompi pala

Olet nyt käynyt esimerkkien avulla läpi kaikki 6502-prosessorin konekielen käskyt. Esimerkit ovat olleet varsin lyhyitä. Ne ovat kukin vuorollaan havainnollistaneet yhtä konekielen toimintoa. Monen käskyn yhteydessä olet varmaan havainnut yhtäläisyyksiä Basic-kielen kanssa. JSR vastaa täysin Basicin GOSUBia jne.

Laajojen ohjelmien tekeminen konekielellä on myös varsin saman kaltaista kuin Basicilla. Suinpäin ei kannata ryhtyä vääntämään koodia vaan ohjelma on suunniteltava hyvin etukäteen.

Konekielellä etukäteissuunnittelu on jopa tärkeämpää. Basicilla voit aina muuttaa ohjelmaasi, tarkistaa välillä sen toimintaa, voit helposti lisätä ohjelma-riivejä ja toimintoja keskellekin ohjelmaasi.

Konekieli on siinä suhteessa armoton. Ohjelman testaaminen on hankalampaa, virhetilanteet johtavat yleensä koneen lukkiutumiseen, väleihin lisääminen ei ole niin yksinkertaista.

Konekielellä ohjelmoitaessa on ohjelman rakenteen oltava selvillä jo ennen ohjelmointia, jotta koodia tuottaessasi voit keskittyä sen virheettömyyteen.

Konekielellä sinun on itse tiedettävä monia asioita, jotka kone hoitaa puolestasi Basicia käytettäessä. On muistettava koneen osoitteita, oman ohjelmasi osoitteita, koodeja. Kynä ja paperi ovat konekieliohjelmoijan välttämättömät perustyökalut.

### Ohjelmointia paperilla

Hyvä ohjelman valmistelu lähtee siitä, että kirjaat paperille ne toiminnot, jotka haluat ohjelmasi suorittavan. Jos esimerkiksi teet pokeri-ohjelman voisivat toimintojen kuvaukset olla vaikkapa seuraavat:

- ohjelma käynnistyy napin painalluksella
- jaa viisi korttia kahteen riviin kuvaruudulle, koneen kortit selkäpuoli ylöspäin, pelaajan kortit näkyviin
- hae näppäimistöltä niiden korttien numerot, jotka pelaaja haluaa vaihtaa. Arvo niihin uudet kortit
- arvioi koneen kortit ja vaihda tarvittaessa
- arvioi kortit ja määrittele voittaja
- mene takaisin alkuun

Tällaisesta toimintojen kuvauksesta on jo helpompi lähteä ohjelmoimaan. Ohjelma on paloiteltu pienempiin osatehtäviin, jotka on yksi kerrallaan paljon helpompi hallita.

Ylläoleva toimintakuvaus on silti aivan liian puutteellinen ohjelman perustaksi. Sitä on voimakkaasti tarkennettava, ennen kuin sen pohjalta kannattaa lähteä ohjelmoimaan. Lisäksi siitä puuttuu pelipanosten käyttö, joten peli ei tällaisenaan olisi kovin mielekäskään.

Toimintaluettelon rivit on hyvä numeroida. Silloin voi riviltä toiselle tehdä hyppyjä, aivan kuin ohjelmoisit Basicilla. Jätä väleihin myös riittävästi käyttämättömiä rivinumeroita, jotta toimintoja voisi lisäillä väleihin. Ei ole lainkaan huono ajatus tehdä ohjelma ensin Basicilla, siten voi tarkistaa ohjelman mielekkyyden.

Ennen ohjelmoinnin alkua on vielä suoritettava muutamia valmisteluja. Ensinnäkin päätettävä, mistä osoitteesta ohjelman aloittaa. Muistiavaruudessa on monia eri kokoisia konekieliohjelmille soveliaita paikkoja. Niistä kannattaa valita ”rauhallisin” riittävän pitkä alue.

Sen jälkeen on arvioitava ohjelman tarvitsemien ”muuttujien” ja vakiotietojen viemä tila ja päätettävä mihin ne sijoittaa. Kynä ja lehtiö on varattava käden ulottuville ja edellä mainitut asiat on merkittävä muistiin. Vakiotietoja voivat olla esimerkiksi kuvaruudulle tietyissä tilanteissa tulostettavat viestit ja muuttujia voivat olla esimerkiksi pistelaskurit.

## Assembler apulaisena

Konekieliohjelmoinnissa käytetään apuna assembler-ohjelmaa. Ainoastaan mitättömän pienet ohjelman pätkät kannattaa koodata käsin taulukoiden avulla. Ohjelmaa ei useinkaan kannata lähteä kirjoittamaan suoraan tietokoneelle vaan parempi tapa on kirjoittaa käskyt ensin paperille ja sitten vasta assembler-ohjelmalle. Ohjelmaasi tulee joka tapauksessa muutoksia ja lisäyksiä.

Jos työskentelet kehittyneemmällä assembler-ohjelmalla, jossa ohjelmakäskyjen syöttö tapahtuu erillisenä tekstieditorin (tekstinkäsittelyohjelman) avulla ja vasta sen jälkeen suoritetaan ohjelman kääntäminen konekielen koodeiksi, voit jättää paperivaiheen pois, varsinkin, jos tekstieditori on hyvä ja sallii ohjelmarivien helpon korjailun ja lisäämisen.

Ohjelmoidessasi käytät kynää ja paperia kuitenkin merkitäksesi muistiin käyttämiäsi muuttujien ja vakioiden osoitteet, eteenpäin suuntautuvien hyppyjen lähtökäskyn ja oletetun kohdekäskyn osoitteet jne.

Ohjelmat eivät koskaan ala suoraan määrittelemiesi toimintojen toteutuksella vaan ohjelmissa on poikkeuksetta aloitusrutiini, joka pohjustaa ohjelman suorituksen. Muuttujille on annettava alkuarvot ja kuvaruudulle on tulostettava alkutekstit.

Kun ohjelmasi on valmis, joko suoraan muistiin ohjelmoituna tai erillisestä tekstitiedostosta käännettynä, voit olla varma siitä, ettei se toimi sellaisenaan. Muista siis tallentaa ohjelma ennen kuin yrität ensimmäistäkään kertaa ajaa sitä. Ohjelma "kolahtaa" varmasti jossain kohdassa ja kone lukkiutuu tai ohjelma käyttäytyy aivan sekopäisesti.

Voit etsiä virheitä monella tavalla. Ohjelman virhetoiminnan laatu ja sen ajoitus ohjelman kuluessa antavat vihjeitä siitä, missä kohtaa virhe sijaitsee. Ensin voi tutkia omaa assembler-listausta tältä kohtaa ja yrittää käskyjä tarkistamalla selvittää syytä virheeseen.

Useissa kehittyneissä monitori/assembler-ohjelmissa voit sijoittaa ohjelmaan keskeytyskohtia, jolloin konekielisen ohjelmasi suoritus pysähtyy valitsemiisi kohtiin ja rekisterien sisällöt ilmestyvät ruudulle, tai voit suorittaa ohjelmaasi jopa käsky kerrallaan.

Jos kuitenkin työskentelet yksinkertaisemmilla välineillä, voit esimerkiksi sijoittaa ohjelmaan epäilyttävään kohtaan RTS-käskyn (\$60), jolloin ohjelman suoritus pysähtyy ja kontrolli palaa takaisin joko Basic-tasolle tai assembler-ohjelmaasi riippuen siitä, mistä olit konekielisen ohjelman käynnistänyt. Muista kokeilun jälkeen palauttaa RTS-käskyn paikalle siinä ollut käskykoodi takaisin. Siirtelemällä RTS-käskyn paikkaa voit paikallistaa ohjelmassasi piilevät virheet.

## Pistelaskuri

Havainnollistaaksemme suuremman konekielisen ohjelman tekoa rakennamme nyt hieman pidemmän ohjelman. Se pitää yllä kuusinumeroista pistelaskuria kuvaruudulla ja lisää siihen lukuja näppäimiä painettaessa.

Ohjelmassa käytetään Commodore 64:n kuvaruutuosoitteita, mutta se on osoitteita muuttamalla sovitettavissa myös muihin mikroihin. Osoitteet löytyvät omaa mikroasi käsittelevästä luvusta kirjan lopusta.

Tämä on jo sen verran monimutkaisempi tehtävä, ettei sen suorittamiseksi kannata lähteä heti pinoamaan konekielen käskyjä. Ensin on mietittävä, mitä ohjelman itse asiassa on tehtävä. Paras tapa on kirjata kaikki pienet toimintapalaset paperille peräkkäin ja numeroida ne. Ihmisen on helpompi ajatella yhtä pientä toimintoa kerrallaan kuin yrittää pitää mielessään kovin laajaa kokonaisuutta. Paperihahmottelussa paljastuu usein jo ensimmäiset virheetkin: usein ei ensimmäinen mieleen tuleva ratkaisutapa johda järkevään tulokseen.



## Konekielinen ohjelmointi

---

Seuraavassa on pistelaskuriohjelmamme toiminta paloitetu pariksikymmeneksi erilliseksi tehtäväksi.

Alkurutiini:

- 1) nollaa pistenäyttö kuvaruudun muistipaikoissa 1054-1059 ja pistelaskuri apurekistereineen, 820-825
- 2) aseta merkkiväri mustaksi
- 3) aseta D-lippu kymmenjärjestelmän laskuja varten

Pääohjelma:

- 4) onko näppäintä painettu?
- 5) jos on, onko se 1, 2, 3 vai 4
- 6) aseta lisättävä luku muistipaikkoihin 823 (vähemmän merkitsevä tavu) ja 824. Näppäin 1 = 1, 2 = 10, 3 = 100 ja 4 = 1000.
- 7) hyppää lisäysrutiiniin, (10)
- 8) hyppää siirto kuvaruudulle -rutiiniin, (14)
- 9) pääohjelman alkuun (päättymätön silmukka), (4)

Luvun lisäys (aliohjelma):

- 10) hae pistelaskurin vähiten merkitsevä tavu
- 11) lisää siihen mp 823, talleta tulos takaisin
- 12) hae 2. tavu, lisää siihen mp 824, talleta tulos takaisin
- 13) hae 3. tavu, lisää \$00 (ja muistinumero), talleta takaisin RTS

Siirto ruudulle (aliohjelma):

- 14) hae vähiten merkitsevä tavu (tavussa on kaksi numeroa)
- 15) jaa kahtia, muuta kumpikin numero vastaavaksi kuvaruutukoodiksi ja vie ruudulle
- 16) hae seuraava tavu, jaa kahtia, vie ruudulle
- 17) hae viimeinen tavu, jaa kahtia, vie ruudulle RTS

Tällaisesta toimintakuvauksesta selviää helposti, onko kaikki oleellinen mukana. Silmäilemällä tätä "algoritmiämme" voimme ikäänkuin ajaa ohjelman mielessämme.

Tässä vaiheessa voimme vielä helposti lisätä puuttuvia toimintoja ja siirrellä niiden tapahtumapaikkaa. Huomaamme esimerkiksi, että luvun lisäyksestä voimme tehdä silmukan sen sijaan, että peräjälkeen käsittelisimme jokaisen tavun eri käskyillä.

Samaa menetelmää voidaan noudattaa myös ruudulle siirrossa. Käytännössä se tarkoittaa, että asetamme indeksirekisteriin tavujen lukumäärän miinus yksi, koska kolme tavua tulee käsiteltyksi seuraavasti:  $X=2$ ,  $X=1$  ja  $X=0$ . Kun lukua kirjoitetaan muistiin, käytämme indeksoitua osoitusmuotoa. Silloin vähennettä-

essä indeksirekisterin arvoa, muuttuu lopullinen osoite sitä mukaa kuin lukuja käsittelemme.

Nyt meillä on koossa järkevältä tuntuva malli ohjelmallemme. Se on osittain vielä ylimalkainen, mutta tässä tapauksessa jo aivan riittävä ohjelmoinnin perustaksi. Jos ohjelmamme sisältäisi paljon haarautumisia, olisi malli tehtävä paljon tarkemmaksi.

Lähdetään listan pohjalta kehittämään konekielistä ohjelmaamme. Käskyriivien perässä olevia kommentteja tarkkaillen voit seurata kunkin erillisen toiminnon paloittumista käskyiksi. Kommenteissa esiintyvät numerot viittaavat paperihahmotelmaamme riveihin.

## Alkurutiini:

- \$0350 LDA #\$00 (akkuun nolla, jolla nollataan muistipaikat)
- \$0352 LDX #\$06 (näyttö on kuusi tavua pitkä, 1)
- \$0354 STA \$0333,X (nollataan samalla laskuri ja muut tavut, 1)
- \$0357 STA \$D81D (merkin väri mustaksi, CBM 64:ssä värikoodi on nolla ja värimuistin alkuosoite 55296, 2)
- \$035A DEX (seuraava tavu)
- \$035B BNE \$0354 (oliko tavuja jäljellä?)
- \$035D SED (aseta desimaalilippu ykköseksi, 3)

## Pääohjelma:

- \$035E JSR \$03AC (nollaa näyttötavut)
- \$0361 JSR \$FFE4 (GETIN-rutiini CBM-64:ssä palauttaa akussa painetun näppäimen ASCII-koodin. Jos näppäintä ei painettu, on akku=0, hahmotelmarivi 4)
- \$0364 BEQ \$0361 (jos nolla, ei näppäintä painettu, palataan 4)
- \$0366 LDX #\$00
- \$0368 CMP #\$31 (oliko ykkönen, 5)
- \$036A BNE \$0371 (jos ei, hyppää eteenpäin, 5)
- \$036C LDA #\$01 (lisättävä luku on yksi, 5)
- \$036E JMP \$036D (hyppää sijoittamaan muistiin)
- \$0371 CMP #\$32 (oliko kakkonen, 5)
- \$0373 BNE \$037A (jos ei, hyppää eteenpäin, 5)
- \$0375 LDA #\$10 (lisättävä luku on 10)
- \$0377 JMP \$038D (hyppää sijoittamaan muistiin)
- \$037A CMP #\$33 (oliko kolmonen, 5)
- \$037C BNE \$0385 (jos ei, hyppää eteenpäin)
- \$037E LDX #\$01 (lisättävä luku on 100)
- \$0380 LDA #\$00 (joten X:ään ykkönen, akkuun 0)
- \$0382 JMP \$038D (hyppää sijoittamaan muistiin)
- \$0385 CMP #\$34 (oliko nelonen, 5)
- \$0387 BNE \$0361 (väärä näppäinpainallus, palaa takaisin)

|        |            |                                      |
|--------|------------|--------------------------------------|
| \$0389 | LDX #\$10  | (lisättävä luku oli 1000)            |
| \$038B | LDA #\$00  | (joten X:ään 10 ja akkuun 00)        |
| \$038D | STA \$0339 | (sijoita lisättävä luku paikoilleen) |
| \$0390 | STX \$0338 |                                      |
| \$0393 | JSR \$039C | (hyppää lisäysrutiiniin, 7)          |
| \$0396 | JSR \$03AC | (hyppää siirtorutiiniin, 8)          |
| \$0399 | JMP \$0361 | (palaa pääohjelman alkuun, 9)        |

## Luvun lisäys:

|        |              |                                                 |
|--------|--------------|-------------------------------------------------|
| \$039C | CLC          | (nollaa Carry ennen yhteenlaskua)               |
| \$039D | LDX #\$03    | (kolme tavua: X = 3, 2, 1)                      |
| \$039F | LDA \$0333,X | (hae pistelaskurin vähiten merkitsevä tavu, 10) |
| \$03A2 | ADC \$0336,X | (lisää tallennettu luku, 11)                    |
| \$03A5 | STA \$0333,X | (vie tulos laskuriin, 11)                       |
| \$03A8 | DEX          | (seuraava tavu, 13)                             |
| \$03A9 | BNE \$039F   | (loppuivatko tavut)                             |
| \$03AB | RTS          | (loppuivat, palataan)                           |

## Siirto ruudulle:

|        |              |                                                                         |
|--------|--------------|-------------------------------------------------------------------------|
| \$03AC | LDX #\$03    | (ladataan X-rekisteriin tavulaskuri, X=2,1,0)                           |
| \$03AE | LDY #\$06    | (Y-rekisteriin numerolaskuri ruudulle tulostusta varten, Y=6,5,4,3,2,1) |
| \$03B0 | LDA \$0333,X | (hae akkuun kaksi numeroa sisältävä tavu)                               |
| \$03B3 | AND #\$0F    | (nollaa neljä ylintä bittiä, jolloin alempi BCD-luku jää akkuun)        |
| \$03B5 | CLC          | (nollaa Carry yhteenlaskua varten)                                      |
| \$03B6 | ADC #\$30    | (muuta luku kuvaruutukoodiksi, CBM 64: luku 1 on koodi \$31)            |
| \$03B8 | STA \$041D,Y | (sijoita koodi kuvaruudulle)                                            |
| \$03BB | DEY          | (seuraava paikka kuvaruudulla)                                          |
| \$03BC | LDA \$0333,X | (lataa sama tavu uudestaan)                                             |
| \$03BF | LSR          | (siirrä tällä kerralla neljä)                                           |
| \$03C0 | LSR          | (ylintä bittiä alempien neljän)                                         |
| \$03C1 | LSR          | (bitin paikalle)                                                        |
| \$03C2 | LSR          |                                                                         |
| \$03C3 | CLC          | (nollaa carry yhteenlaskua varten)                                      |
| \$03C4 | ADC #\$30    | (muuta kuvaruutukoodiksi)                                               |
| \$03C6 | STA \$041D,Y | (vie kuvaruudulle)                                                      |
| \$03C9 | DEX          | (seuraava tavu)                                                         |
| \$03CA | DEY          | (seuraava kuvaruutupaikka)                                              |
| \$03CB | BNE \$03B0   | (jos numeroita jäljellä, palaa takaisin)                                |
| \$03CD | RTS          | (muutoin palaa aliohjelmasta)                                           |

Basicilla äskeinen tehtävä olisi ollut varsin helppo ja lyhyt. Basicissahan on suoraan käytettävissä pidempien lukujen laskutoimitukset yhdellä käskyllä ja luvun voi printata ruudulle myös yhdellä käskyllä. Basicilla esimerkki olisi ollut toteutettavissa seuraavalla tavalla:

```
10  P%=0:L%=0
20  P%=P%+L%: L%=0
30  PRINT "<HOME>"P%
40  GET A$: IF A$="" THEN 40
50  IF A$="1" THEN L%=1
60  IF A$="2" THEN L%=10
70  IF A$="3" THEN L%=100
80  IF A$="4" THEN L%=1000
90  GOTO 20
```

Rivillä 30 esiintyvä <HOME> merkitsee kursorin siirtoa kuvaruudun yläreunaan tyhjentämättä ruutua.

Konekielisessä esimerkissämme alkurutiini, luvun lisäys ja siirto ruudulle voivat olla osia omaa konekielistä ohjelmaasi. Pääohjelma on esimerkissä mukana vain sitomassa ne toimivaksi kokonaisuudeksi. Oma ohjelmasi sisältää erilaisen pääohjelman, mutta luvun lisäyksessä voit käyttää annettua rutiinia sijoittamalla siellä muistipaikkoihin 823 ja 824 lisättävän luvun ja kutsumalla lisäysrutiinia osoitteesta \$304C (tai siitä osoitteesta, johon olet kyseisen rutiinin sijoittanut ohjelmaa tehdessäsi). Samaten uuden arvon saat siirrettyä kuvaruudulle omassa ohjelmassasi käskyllä JSR \$305C, mikäli olet lisännyt rutiinin omaan ohjelmaasi.

## Tarkastaja

Kirjan lopussa olevassa liitteessä on pieni konekielinen ohjelma, Tarkastaja. Se on tarkoitettu CBM-64:n ja VIC-20:n omistajille, jotka voivat sen avulla helpottaa listausten syöttöä koneeseensa.

Commodoren ohjelmalistauksissa on jokaisella rivillä viimeisenä REM-lause, jossa on kaksi heksanumeroa. Ne muodostavat tarkistussumman kyseisen rivin kirjaimista ja muista merkeistä.

Tarkastaja on ohjelma, joka toimii "taustalla" siten, että jokaisen ohjelmarivin syötön päättyessä RETURN-näppäimen painallukseen, se laskee sisäänsyötetyn rivin tarkistussumman ja tulostaa sen kuvaruudun ylälaitaan. Vertaamalla kuvaruudulla olevaa tarkistussummaa listauksessa olevaan voi varmistua siitä, että rivi on syötetty oikein.

## Koukku

Tarkastaja on konekielinen ohjelma. Tutkimme sen CBM-64 -versiota toisena esimerkkinä pidemmästä konekielisestä ohjelmasta.

Tarkastaja ei saa häiritä laitteen muita toimintoja. Se on toteutettu siten, että se ei ole toiminnassa kaiken aikaa, vaan se on "koukutettu" kiinni käyttöjärjestelmään, sen CHRIN eli merkinhakurutiiniin. Joka kerta, kun näppäimistöltä haetaan tieto näppäimen painalluksesta, suorittaa CBM-64 CHRIN-rutiinin, jonka alkuosoitteen se löytää osoitteista \$0324 ja \$0325. Tavallisesti näistä muistipaikoista löytyy osoite \$F157, jossa varsinainen CHRIN sijaitsee.

Teemme siten, että sijoitamme näihin muistipaikkoihin (\$0324 ja \$0325) oman konekielisen pääohjelmamme alkuosoitteen, joka on \$02B2. Seuraavalla rutiinilla Tarkastaja otetaan käyttöön ennen Basic-ohjelman syötön aloittamista. Se käynnistetään käskyllä SYS 679.

|        |            |                                                       |
|--------|------------|-------------------------------------------------------|
| \$02A7 | LDA #02    | (lataa akkuun oman ohjelmamme osoitteen ylätavu)      |
| \$02A9 | STA \$0325 | (vie se osoitteeseen \$0325)                          |
| \$02AC | LDA #\$B2  | (lataa akkuun oman ohjelmamme alkuosoitteen alataavu) |
| \$02AE | STA \$0324 | (vie se osoitteeseen \$0324)                          |
| \$02B1 | RTS        | ('koukku' on asetettu, palataan Basiciin)             |

Nyt olemme saaneet käyttöjärjestelmän hallintaamme: aina merkkiä hakiesaan käyttöjärjestelmä siirtyykin omaan konekieliseen ohjelmaamme.

Koska käyttöjärjestelmän tarkoituksena on suorittaa CHRIN-rutiini, emmekä halua häiritä sen toimintaa, on ensimmäinen tehtävä omassa ohjelmassamme käydä suorittamassa CHRIN-rutiini. Teemme sen käskyllä

**\$02B2 JSR \$F157**

Kun rutiini on hakenut näppäimistöltä merkin, palaa se takaisin omaan ohjelmaamme. CHRIN-rutiinin ominaisuuksiin kuuluu, että haettu näppäinkoodi sijaitsee akkurekisterissä. Indeksirekistereissä sijaitsee rutiinin jälkeen tietoja mm. merkkien määrästä rivillä jne., mutta ne eivät meitä kiinnosta, kunhan emme puutu indeksirekistereiden sisältöön millään tavalla.

Kun tällä tavalla on mennyt "kutsumattomaksi vieraaksi" käyttöjärjestelmän aliohjelmaan, on syytä olla erittäin varovainen: paikat eli rekisterit on syytä jättää täsmälleen samanlaisiksi kuin ne aluperin olivatkin. Sen vuoksi on jo tässä vaiheessa syytä tallentaa kaikki ne rekisterit, joiden sisältöä oma ohjelmamme muuttaa, jotta niiden alkuarvot voidaan viimeisenä tehtävänä jälleen palauttaa ennen paluuta käyttöjärjestelmään. Sen vuoksi tilarekisteri siirretään pinoon ja akku näppäintietoineen kopioidaan muistipaikkaan \$FC.

|        |          |                                 |
|--------|----------|---------------------------------|
| \$02B5 | STA \$FC | (tallenna akku)                 |
| \$02B7 | PHP      | (tallenna tilarekisteri pinoon) |

Ohjelman loppuosan toiminta käy parhaiten selville assemblylistauksen kommentteista. Akussa on edelleen painetun näppäimen koodi.

**\$02B8 CMP #\$20** (oliko välilyönti? Välilyöntejä ei lasketa tarkistus-

|        |            |                                                                                                                                                                                                                          |
|--------|------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|        |            | summaan)                                                                                                                                                                                                                 |
| \$02BA | BEQ \$02EB | (jos oli, siirry pois Tarkastajasta)                                                                                                                                                                                     |
| \$02BC | CMP #\$0D  | (oliko RETURN?)                                                                                                                                                                                                          |
| \$02BE | BNE \$02E6 | (jos ei ollut, siirrytään laskemaan tarkistussummaa, muuten jatketaan siirtämällä laskettu tarkistussumma ruudulle)                                                                                                      |
| \$02C0 | LDA \$FD   | (tarkistussummaa pidetään muistipaikassa \$FD)                                                                                                                                                                           |
| \$02C2 | AND #\$0F  | (maskataan neljä ylintä bittiä pois, jotta voimme käsitellä neljän alimman bitin heksanumeroa)                                                                                                                           |
| \$02C4 | ORA #\$B0  | (asetetaan 5., 6. ja 8. bitti ykkösiksi. Ensimmäiset muuttavat heksanumeron vastaavaksi ASCII-koodiksi kuvaruututulostusta varten ja 8. bitti määrittelee tulostuksen käänteiseksi merkiksi, vaalea tummalla pohjalalla) |
| \$02C6 | JSR \$0334 | (muutetaan kirjaimet A-F vastaaviksi kuvaruutukoodiksi)                                                                                                                                                                  |
| \$02C9 | STA \$042A | (siirretään heksanumero kuvaruudulle)                                                                                                                                                                                    |
| \$02CC | LDA \$FD   | (haetaan tarkistussumma uudestaan)                                                                                                                                                                                       |
| \$02CE | LSR        | (ja siirretään tällä kertaa)                                                                                                                                                                                             |
| \$02CF | LSR        | (neljä ylintä bittiä)                                                                                                                                                                                                    |
| \$02D0 | LSR        | (alimmaisten bittien paikalle muok-)                                                                                                                                                                                     |
| \$02D1 | LSR        | (kaamista ja ruudulle siirtoa varten)                                                                                                                                                                                    |
| \$02D2 | ORA #\$B0  | (jälleen käänteiset merkit ja ASCII-koodi)                                                                                                                                                                               |
| \$02D4 | JSR \$0334 | (kirjaimet A-F kuvaruutukoodiksi)                                                                                                                                                                                        |
| \$02D7 | STA \$0429 | (toinen heksaluku ruudulle)                                                                                                                                                                                              |
| \$02DA | LDA #\$00  | (taustaväri mustaksi, koska käänteiset merkit)                                                                                                                                                                           |
| \$02DC | STA \$D829 | (merkkiväri värimuistiin)                                                                                                                                                                                                |
| \$02DF | STA \$D82A | (vastaaville paikoille)                                                                                                                                                                                                  |
| \$02E2 | LDA #\$00  | (pakotettu hyppy uuteen osoitteeseen, loppurutiiniin,)                                                                                                                                                                   |
| \$02E4 | BEQ \$02E9 | (koska ehto aina tosi)                                                                                                                                                                                                   |

Seuraavaan käskyyn hypättiin, jos näppäimistöltä haettu merkki ei ollut väliönti eikä RETURN. Näppäimen arvo lasketaan yhteen kertyneen tarkastussumman kanssa.

|        |          |                                                                                                                                                                            |
|--------|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$02E6 | CLC      | (yhteenlasku alkaa muistinumeron nollauksella)                                                                                                                             |
| \$02E7 | ADC \$FD | (akussa oleva koodi lisätään tarkistussummaan)                                                                                                                             |
| \$02E9 | STA \$FD | (tulos viedään takaisin paikalleen. Huomaa, että samalla käskyllä hoidetaan myös tarkistussumman nollaus ruudulle siirron jälkeen: tähän osoitteeseen hypätään \$02E4:stä) |
| \$02EB | LDA \$FC | (palautetaan akun alkuperäinen sisältö)                                                                                                                                    |
| \$02ED | PLP      | (palautetaan tilarekisterin sisältö)                                                                                                                                       |

**\$02EE RTS (palataan takaisin Basic-tulkkiin)**

Ohjelmaan kuuluu vielä seuraava aliohjelma osoitteissa \$0334-\$033B. Se muuttaa heksanumeron kuvaruutukoodiksi siinä tapauksessa, että se on kirjain A-F. Numeroille 0-9 rutiini ei tee mitään.

**\$0334 CMP #\$BA (oliko heksanumero pienempi kuin A? Huomaa, että bitit 5, 6, ja 8 asetettiin, joten A:ta edeltää B)**

**\$0336 BCC \$033B (jos oli, palataan rutiinista)**

**\$0338 SEC (asetetaan Carry-lippu vähennyslaskua varten)**

**\$0339 SBC #\$39 (vähennetään koodista \$39, jolloin kirjaimet A-F osuvat oikeille kuvaruutukoodeille, 1-6)**

**\$033B RTS (palataan aliohjelmasta)**

Tarkastajassa on kaksi turhaa käskyä, jotka ovat mukana selvyiden vuoksi: osoitteessa \$02E2 ei akkua tarvitsisi nollata, koska se on hetkeä aiemmin jo nolattu ja osoitteessa \$0338 ei Carry-lippua tarvitsisi asettaa, koska vähennyslaskuun mennään vain, jos C-lippu on jo asettunut (edellinen käsky on BCC).

### Assemblerin käyttö

Assembler-kääntäjällä työskentely muistuttaa hyvin paljon muilla ohjelmointikielillä työskentelyä. Lukuunottamatta aivan yksinkertaisimpia assemblereita, tapahtuu assembly-kielellä ohjelmointi seuraavasti:

1) Ohjelma suunnitellaan ennakkoon, se jaetaan osatehtäviin, joista kullekin määritellään suoritettavat toiminnot. Jos et tätä suunnittelua tee paperille, toimivat aivosi kuitenkin samaan tapaan.

2) Assembler-ohjelma käynnistetään, valitaan ohjauskäskyllä uuden ohjelman kirjoitus, tai jatkettaessa aiempaa ohjelmointityötä, haetaan kasetilta tai levy-muistista jo kirjoitettu ohjelman osa ja jatketaan sen perään.

3) Kun koko ohjelma on kirjoitettu, yksi käsky argumentteineen ja kommentteineen aina yhdelle riville, on meillä koossa niin kutsuttu lähdeohjelma (source). Se on pelkkää kirjainkoodia, joten mikroprosessoria ei voi pistää suorittamaan sitä.

4) Annetaan assembler-ohjelmalle ohjauskäsky, joka kehottaa sitä kääntämään lähdeohjelmamme konekielikoodiksi (object). Tämän assembler-ohjelma sijoittaa joko suoraan muistiin oikeille paikoilleen tai levykeasemaa käytettäessä usein levykkeelle ohjelmatiedostoksi. Monet assembler-ohjelmat tulostavat tässä vaiheessa täydellisen listauksen käännetyistä ohjelmasta. Listaus sisältää kaiken sen, mitä olemme itsekkin ohjelmaan kirjoittaneet, mutta lisäksi siihen kuuluu jokaiselle riville rivinumero, rivillä sijaitsevan käskyn tuleva osoite muistissa, käsky ja argumentti heksalukuina. Listauksen loppuun assembler-ohjelma liittää vielä luettelon käytetyistä "nimistä" eli leimoista ja mahdolliset virheilmoitukset.

5) Jos koodi sijoitettiin muistiin, on se suoraan ajettavissa Basicissa SYS-käs-



kyllä tai assembler-ohjelman yhteydessä yleensä kirjainkäskyllä G. Useissa assembler-ohjelmissa kirjainkäskyt merkitsevät samoja asioita, muttei välttämättä. Se riippuu assembler-ohjelman tehneestä ohjelmiojasta.

6) Kun ohjelma on käynnistetty se ei toimi. Vähänkin pidemmällä ohjelmalla näin käy 99:ssä tapauksia. Hyvin harvoinhan Basic-ohjelmataan toimivat heti täydellisesti. Sen jälkeen alkaakin ohjelman korjausvaihe, joka muodostaa usein merkittävän osan koko ohjelmointityöstä.

Eri tasoisilla assemblereilla työskentely on erilaista. Yksinkertaiset assemblerit sijoittavat konekielikoodit muistiin samalla, kun kirjoitat käskyn ruudulle ja painat RETURNia. Nämä assemblerit eivät yleensä anna ohjelmasta listausta vaan se on otettava muulla tavoin, esimerkiksi disassemblerilla.

Disassembler on ohjelma, joka toimii täysin päinvastaisesti kuin assembler. Disassembler tutkii muistipaikan sisällön, hakee taulukostaan sieltä löytyvää koodia vastaavan kolmikirjaimisen käskysanan ja tulostaa koko ohjelman selväkielisin käskylyhentein ruudulle (ja kirjoittimelle).

Disassembler on välttämätön työkalu tutkittaessa ohjelmia, joista ei ole olemassa listausta. Disassembler automatisoi kaiken taulukkoruljanssin koodien etsimiseen. Taulukoitakin tarvitaan. Jos näkee lyhyen ohjelman pätkän heksalistsauksena tai Basicin DATA-lauseissa, ei aina ole nopein tapa kaivaa esiin tietokonetta, disassembleria ja naputtaa koodeja koneeseen. Taulukosta parikymmentä käskyä löytyy varmasti nopeammin.

Lähes aina assembler-ohjelmat sisältävät myös disassembler-toiminnon, eli disassembler sisältyy assembleriin.

## Monitorit

Olet varmaan kuullut puhuttavan myös konekielimonitoreista. Usein kielenkäyttö assemblerin ja monitorin osalta lipsuu ja niillä tarkoitetaan samaa asiaa. Monitoriohjelma on kuitenkin paljon suppeampi kuin assembler. Monitorilla voit monitoroida eli tutkia koneesi muistia. Monitori pystyy usein vain näyttämään halutun muistipaikan sisällön ja korvaamaan sen antamallasi luvulla. Monitorien ominaisuuksiin kuuluu myös muistialueiden siirto paikasta toiseen.

Mutta jos ohjelma pystyy kääntämään assembly-kieltä konekielikoodiksi, on se jo kunnan assembler-ohjelma ja sitä tulisi myös kutsua assembleriksi eikä monitoriksi. Tekstitiedostosta kääntävää assembleria sanotaan assembler-kääntäjäksi.

Assembler-ohjelma sen sijaan voi laajimmillaan sisältää kaiken: disassemblerin, monitorin ja debuggerin.

Debugger on ohjelma, jolla konekielisistä ohjelmista etsitään virheitä. Se pystyy lisäämään pötkössä olevien käskyjemme joukkoon pysäytyskäskyjä, jolloin voimme ajaa ohjelmaa pala kerrallaan ja todeta, missä ohjelmalohkossa virhe tapahtuu. Haarukoimalla tarkemmin voi pysäytyskäskyjä sijoittamalla lopulta paikallistaa itse viallisen käskykohdan.

### Eteenpäin

Näiden esimerkkien jälkeen konekielisen ohjelman rakenne ja käskyjen käyttö toistensa yhteydessä alkaa hiljalleen valjeta. Tämä ei kuitenkaan riitä kovinkaan pitkälle koneen käskemisessä. Jokainen tietokone on rakennettu eri tavalla. Kuvaruutumuisti sijaitsee eri paikoissa, peliohjaimet luetaan eri muistipaikoista, värien määrittelyt tehdään eri tavalla. Käyttöjärjestelmät ovat myös erilaiset ja niissä olevat hyödylliset rutiinit, joita voi käyttää omissa konekielisissä ohjelmis- saan sijaitsevat eri kutsuosoitteissa ja jopa toimivat eri tavalla.

Konekieliseen ohjelmointiin tarvitaan siis paljon muutakin tietoa kuin pelkäs- tään konekielen käskyjen ja osoitusmuotojen hallintaa. On tunnettava koneensa rakenne.

Tärkein rakenteellinen tekijä on laitteen muistikartta, joka konekieliohjelmoi- jalle antaa kaiken sen tiedon, mitä käskyjen lisäksi tarvitaan.

Seuraavissa luvuissa käsitellään suosituimpien 6502-mikroprosessoria käyttä- vien kotitietokoneiden muistikarttoja. Tämä on vaihe, jossa lukijat haarautuvat kukin omalle polulleen. Muistikarttoja käsittelevät tiedot eivät ole eri koneille edes osittain yhteisiä, paitsi CBM-64:n ja VIC-20:n tapauksessa, jotka laitteina ovat kuin veljeksiä.

Kukin seuraavista kappaleista antaa tietoja konekieliseen ohjelmointiin yhdel- lä konemerkillä. Ne riittävät varsin pitkälle, mutta harrastuksen edistyessä saattaa lisätiedon tarve yllättää. Sen vuoksi jokaisen kappaleen loppuun on liitetty luet- telo kutakin konemerkkiä koskevasta kirjallisuudesta, joka valitettavasti on useimmissa tapauksissa englannin kielistä.

## Konekielinen ohjelmointi

# 8



# Commodore 64

Konekielinen ohjelmointi on harvoin pelkkää numeron pyörittystä muistissa. Toiminnaltaan mielekäs ohjelma joutuu jatkuvasti kommunikoimaan ympäristönsä kanssa. Merkkejä viedään ruudulle, luetaan näppäimistöä ja ohjaimia, joskus nauhuria tai levyasemaa. Käyttöjärjestelmäkin asettaa omalle ohjelmallemme sellaisen ulkoisen tekijän, joka on otettava huomioon.

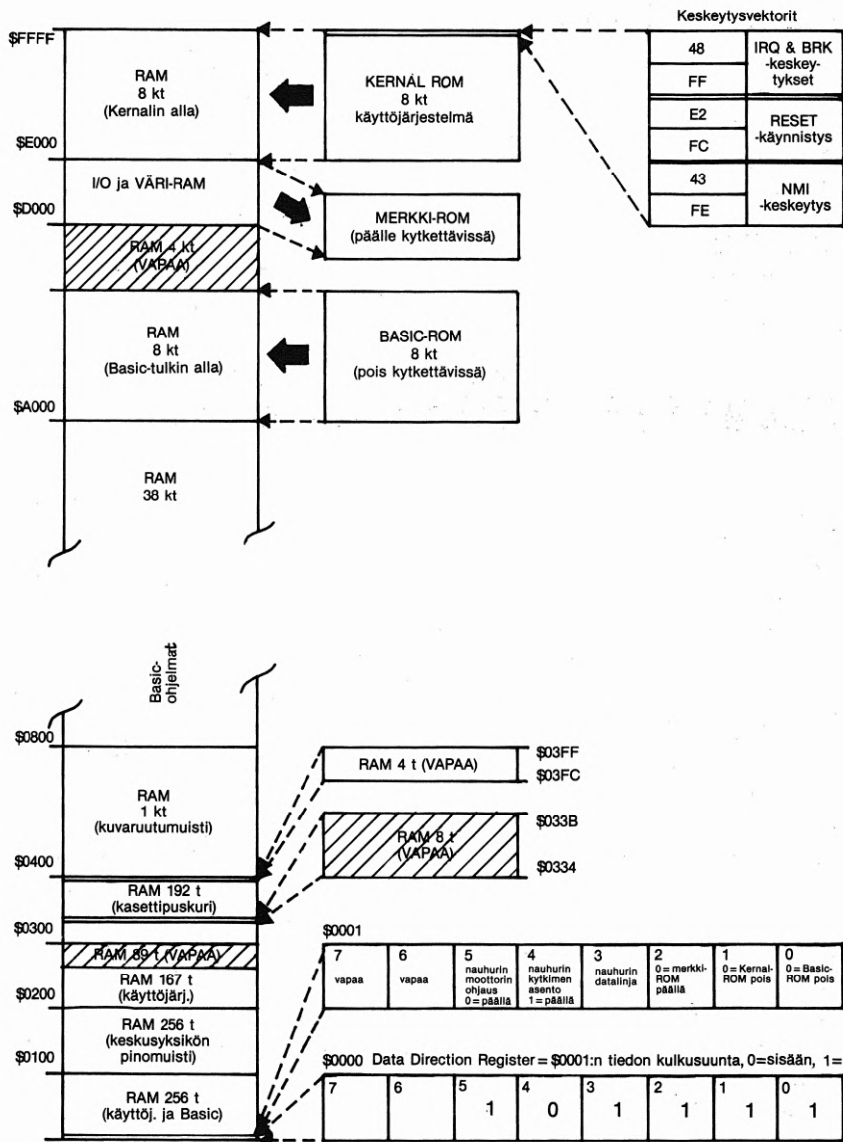
## Muistikartta

Jotta ulkoisia tekijöitä voi ohjelmissaan käyttää hyväksi on tunnettava niiden sijainti muistialueella ja toimintatapa. Muistikartta on havainnollisin tapa selvittää, mitä tietokoneen sisällä oikein tapahtuu.

Oheisessa kuvassa on Commodore 64:n muistikartta, joka ulottuu \$0000:sta \$FFFF:ään eli se sisältää yhteensä 65536 muistipaikkaa. Erikoisin ominaisuus 64:n muistissa on sen kaksijakoisuus: koko 64 kilotavun alue on RAM-muistin (RAM=luku/kirjoitusmuisti) peitossa, mutta siitä huolimatta muistialueella sijaitsevat mm. käyttöjärjestelmä ja Basic-tulkki, jotka ovat ROM-muistia (luku-muistia). Normaaliutilassa keskusyksikkö kommunikoi näiden ROM-muistien ja niiden ulkopuolelle jäävien RAM-muistialueiden kanssa. Osa RAM-muistista jää siis piiloon ROMin alle. Vaihtamalla muistipaikan \$0000 kolmen ensimmäisen bitin arvoa saadaan nämäkin RAM-muistialueet käyttöön kukin erikseen.

Omassa konekieliohjelmassaan voi Basic-tulkin huoletta kytkeä pois päältä, mikäli ei ohjelmassaan käytä Basicin alirutiineja. Käyttöjärjestelmän pois kytkeminen onkin jo mutkikkaampi asia. Käyttöjärjestelmä pitää nimittäin huolen keskusyksikön keskeytyksistä ja keskeytysrutiiniosoitteista. Käyttöjärjestelmää pois kytkettäessä on varmistettava näiden toimintojen jatkuminen.

Merkittävä asia ROMin ja RAMin päällekkäisyydessä on vielä se, että kirjoitet-



Commodore 64:n muistikartta. 64:ssä käytetään 6510-prosessoria ja sen kaksi alinta muistipaikkaa toimivat eri tavalla kuin 6502:ssa. Ne on varattu ohjaamaan ROM-piirejä, jotka voidaan kytkeä pois toiminnasta (Kernal ja Basic) sekä merkkiROMia ja kasettinauhuria.

taessa (STA tai POKE) jotain normaalitilassa ROM-alueelle tieto meneekin ROMin alla olevaan RAM-muistiin ja pysyy siellä. Luettaessa (LDA tai PEEK) näitä osoitteita saadaan arvoksi ROM-muistissa oleva tavu. Alla olevaa RAMia voidaan siis lukea vain, jos vastaava ROM ensin kytketään pois päältä.

## Vapaat alueet

Konekieliohjelmoijalle on ensimmäinen tärkeä asia tietää, missä osoitteissa sijaitsee vapaata tilaa konekieliohjelmien käyttöön. Commodore 64:ssä on normaalitilassakin useita kilotavuja vapaata tilaa, jota kone ei käytä lainkaan. Niihin voi siis tallentaa omia konekieliohjelmiaan ilman, että Basiciltakaan viedään lainkaan tilaa.

Pisin näistä alueista sijaitsee osoitteissa \$C000-\$CFFF (49152-53247), joka on 4 kilotavua pitkä. Lyhyempiä pätkiä sijaitsee osoitteissa \$02A7-\$02FF (679-767) eli 89 tavua, \$0334-\$033B (820-827) eli 8 tavua ja \$03FC-\$03FF (1020-1023) eli 4 tavua.

Neljän tavun pituiseen tilaan ei mahdu juuri muuta kuin tallennettavia lukuja. Tärkeitä ovat kuitenkin nollasivulla sijaitsevat parin tavun mittaiset vapaat paikat, joita voi käyttää epäsuorien indeksoitujen osoitusmuotojen yhteydessä pitämään sisällään lopullista osoitetta. Kaksi tällaista osoitetta mahtuu muisti-paikkoihin \$00FB-\$00FE (251-254).

Nämä olivat siis täysin vapaita paikkoja, joiden käyttäminen ei rajoita koneen muuta toimintaa lainkaan.

Eräs varsin suosittu konekieliohjelmien säilytyspaikka on 64:n kasettipuskuri. Se sijaitsee osoitteissa \$033C-\$03FB (828-1019). Se on vapaata tilaa, mutta ainoastaan, jos kasettiasemaa ei käytetä.

Jos haluaa tehdä pitempiä konekielisiä ohjelmia, on RAM-tilaa tavalla tai toisella "lainattava" koneen muilta toiminnoilta. Suurin lainattavissa oleva alue on se 38 kilotavun tila, missä käyttäjän omat Basic-ohjelmat ovat tallennettuina. Muistipaikoissa \$0037-\$0038 (55-56) sijaitsee Basicin käytössä olevan muistialueen ylärajan osoite. Käyttöjärjestelmä ja Basic-tulkki käyttävät näiden muisti-paikkojen sisältöä hyväkseen sijoittaessaan Basic-ohjelmaa ja sen tuottamia muuttujia muistiin. Voimme käydä muuttamassa tätä osoitetta vaiivikkaa ennen kuin Basic-ohjelmaa syötetään koneeseen tai sen ajon alussa, jolloin saamme huijattua itsellemme vapaata konekielitilaa. Basicista käsin se käy seuraavalla käskyllä:

**(10) POKE 56, PEEK(56)-4\*X: CLR**

jossa X tarkoittaa tällä tavalla ryöstettyjen kilotavujen määrää. Osoitteessa 56 sijaitsee yläpään osoitteen enemmän merkitsevä tavu eli jokainen yksikkö siinä vastaa 256 muistipaikkaa.

Jos menemme vielä pidemmälle tilan varaamisessa, voimme aukaista 8 kilota-

vua RAM-muistia kytkemällä Basic-tulkin sisältävän ROM-muistin pois päältä. Sen voisimme tehdä Basicistakin käsin muuttamalla muistipaikan 1 sisältöä sopivasti, mutta siinä ei olisi mitään mieltä, koska seuraavaa Basic-käskyä ei enää suoritettaisi. Se on sen vuoksi tehtävä konekielillä, esimerkiksi seuraavasti:

|               |                  |                                       |
|---------------|------------------|---------------------------------------|
| <b>\$3000</b> | <b>LDA \$01</b>  | (lataa akkuun muistipaikan 1 sisältö) |
| <b>\$3002</b> | <b>AND #\$FE</b> | (maskaa bitti 0 nolllaksi)            |
| <b>\$3004</b> | <b>STA \$01</b>  | (vie tulos muistipaikkaan 1)          |
| <b>\$3006</b> | <b>...</b>       |                                       |

Basic-tulkin poistamisella saadaan lisätilaa, mutta sen jälkeen ei voi enää käyttää hyväkseen Basicissa sijaitsevia valmisrutiineja. Sama koskee KERNALin eli käyttöjärjestelmän poiskytkemistä, joka tehdään edellisen esimerkin mukaisesti, mutta muuttamalla AND-käskyn argumentiksi \$FD. Samanaikaisesti ne poistetaan argumentilla \$FC.

ROM-muistin poiston jälkeen konekieliohjelmamme on aivan omillaan. Mitään koneen valmisrutiineista ei ole käytettävissä. Ennen käyttöjärjestelmän poiskytkemistä on vielä varmistettava keskeytysten palveluiden jatkuminen, muuten saattaa tulla ongelmia.

Hyvin harvoin konekieliohjelmasi ovat niin pitkiä, ettei niille riittäisi 4 kilotavun täysin vapaa alue lisättynä tarpeellisella määrällä Basic-ohjelmilta lainattua muistia.

## Haetaan näppäimiltä

Konekieliohjelmienkin on haettava ohjausta käyttäjältään. Peleissä se useimmiten hoituu peliohjaimen välityksellä, mutta muissa tilanteissa luonnollisin yhteydenpitoväline on näppäimistö.

Commodore 64:ssä näppäimistöllä on oma pieni puskurinsa. Se tarkoittaa, että näppäimiä painettaessa painetun näppäimen koodi sijoitetaan tietylle muistialueelle ja tietyn rekisterin arvo kasvatetaan yhdellä. Tämä tapahtuu, vaikkei näppäimistöä luettaisikaan sillä hetkellä. Puskurin pituus on kymmenen merkkiä ja se sijaitsee osoitteissa \$0277-\$0280 (631-640). Puskurissa olevien painallusten lukumäärä voidaan lukea muistipaikasta \$00C6 (198).

Luettaessa näppäimistöä puskurin kautta käydään ensin tarkistamassa, onko siellä näppäilyjä odottamassa. Jos muistipaikan 198 sisältö on nolla, ei tavaraa ole noudettavissa. Jos se on suurempi, voidaan ensimmäinen arvo hakea osoitteesta 631. On muistettava vähentää laskurin arvoa yhdellä (mp 198) ja samaten siirtää puskuria.

Jos ohjelmaa kiinnostaa ainoastaan tällä hetkellä painettava näppäin eikä puskuria tarvita, voidaan tieto alhaalla olevasta näppäimestä hakea osoitteesta \$00CB (203). Jos siellä on luku 64, ei mitään näppäintä paineta, muuten se sisältää painetun näppäimen koodin. Osoitteesta 197 löytyy viimeksi painetun näppäimen koodi, vaikkei lukuhetkellä mitään näppäintä enää painettaisikaan.

## COMMODORE 64:N NÄPPÄINKOODIT

| näppäin | koodi | näppäin                                                                                | koodi |
|---------|-------|----------------------------------------------------------------------------------------|-------|
| A       | 10    | 6                                                                                      | 19    |
| B       | 28    | 7                                                                                      | 24    |
| C       | 20    | 8                                                                                      | 27    |
| D       | 18    | 9                                                                                      | 32    |
| E       | 14    | 0                                                                                      | 35    |
| F       | 21    | + (—)                                                                                  | 40    |
| G       | 26    | — (=)                                                                                  | 43    |
| H       | 29    | £ (:)                                                                                  | 48    |
| I       | 33    | CLR/HOME                                                                               | 51    |
| J       | 34    | INST/DEL                                                                               | 0     |
| K       | 37    | ←                                                                                      | 57    |
| L       | 42    | @ (Å)                                                                                  | 46    |
| M       | 36    | * (@)                                                                                  | 49    |
| N       | 39    | ↑                                                                                      | 54    |
| O       | 38    | : (Ö)                                                                                  | 45    |
| P       | 41    | ; (Ä)                                                                                  | 50    |
| Q       | 62    | = (;)                                                                                  | 53    |
| R       | 17    | RETURN                                                                                 | 1     |
| S       | 13    | ,                                                                                      | 47    |
| T       | 22    | .                                                                                      | 44    |
| U       | 30    | /                                                                                      | 55    |
| V       | 31    | CRSR  | 7     |
| W       | 9     | CRSR<=>                                                                                | 2     |
| X       | 23    | f1                                                                                     | 4     |
| Y       | 25    | f3                                                                                     | 5     |
| Z       | 12    | f5                                                                                     | 6     |
| 1       | 56    | f7                                                                                     | 3     |
| 2       | 59    | VÄLILYÖNTI                                                                             | 60    |
| 3       | 8     | RUN/STOP                                                                               | 63    |
| 4       | 11    | EI PAINETTU                                                                            | 64    |
| 5       | 16    |                                                                                        |       |

Hakemalla tavu muistipaikasta \$CB (203) saadaan selville, mitä näppäintä parhaillaan painetaan. Jos mitään näppäintä ei paineta, on muistipaikan 203 arvo \$40 (64). Suluissa ovat skandinaavisen näppäimistön merkit.



## Luetaan peliohjainta

Kuusnelosessa on kaksi peliohjainporttia. Ne sijaitsevat osoitteissa \$DC01 (56321) ja \$DC00 (56320). Näiden muistipaikkojen viisi alinta bittiä kuvaa peliohjaimen kytkimiä seuraavasti:

- bitti 4 tulitus
- bitti 3 oikealle
- bitti 2 vasemmalle
- bitti 1 taakse
- bitti 0 eteen

Normaalitilassa bitit ovat ykkösiä. Jos tulitusnäppäintä painetaan, muuttuu bitti 4 nolllaksi, jos sauvaa käännetään vasemmalle muuttuu bitti 2 nolllaksi jne. Jos sauvaa käännetään oikealle eteen, muuttuvat bitit 0 ja 3 nollliksi, jos sauvaa käännetään alas vasemmalle, muuttuvat bitit 1 ja 2 nollliksi jne.

Seuraavaa ohjelmanpätäkää voit käyttää omissa ohjelmissasi testaamaan peliohjainten asentoa ja haarautumaan sen mukaan:

|        |     |        |                                                |
|--------|-----|--------|------------------------------------------------|
| \$3000 | LDA | \$DC01 | (hae peliohjaimen 1 bitit akkuun)              |
| \$3003 | LSR |        | (siirrä C-lippuun ETEEN-bitti)                 |
| \$3004 | BCS | \$3009 | (jos C=1, jatketaan ilman toimenpiteitä)       |
| \$3006 | JSR | \$3100 | (muuten käydään ensin ETEEN-rutiinissa)        |
| \$3009 | LSR |        | (siirrä C-lippuun TAAKSE-bitti)                |
| \$300A | BCS | \$300F | (jos C=1, jatketaan ilman toimenpiteitä)       |
| \$300C | JSR | \$3200 | (muuten käydään ensin TAAKSE-rutiinissa)       |
| \$300F | LSR |        | (siirrä C-lippuun VASEMMALLE-bitti)            |
| \$3010 | BCS | \$3015 | (jos C=1, jatketaan ilman toimenpiteitä)       |
| \$3012 | JSR | \$3300 | (muuten käydään ensin VASEMMALLE-rutiinissa)   |
| \$3015 | LSR |        | (siirrä C-lippuun OIKEALLE-bitti)              |
| \$3016 | BCS | \$301B | (jos C=1, jatketaan ilman toimenpiteitä)       |
| \$3018 | JSR | \$3400 | (muuten käydään ensin OIKEALLE-rutiinissa)     |
| \$301B | LSR |        | (siirrä C-lippuun TULITUS-bitti)               |
| \$301C | BCS | \$3021 | (jos C=1, jatketaan ilman toimenpiteitä)       |
| \$301E | JSR | \$3400 | (muuten käydään ensin TULITUS-rutiinissa)      |
| \$3021 | ... |        | (ohjelma jatkuu...)                            |
| \$3100 | ... |        | (aliohjelma, joka suorittaa siirron eteenpäin) |

Tässä ja seuraavissa aliohjelmissa on akku tallennettava, jotta muutkin ohjausbitit voitaisiin lukea.

|        |     |                                                 |
|--------|-----|-------------------------------------------------|
| \$31FF | RTS | (paluu pääohjelmaan)                            |
| \$3200 | ... | (aliohjelma, joka suorittaa siirron taaksepäin) |
| \$32FF | RTS | (paluu pääohjelmaan)                            |
| \$3300 | ... | (aliohjelma, joka suorittaa siirron vasemmalle) |
| \$33FF | RTS | (paluu pääohjelmaan)                            |

|        |     |                                               |
|--------|-----|-----------------------------------------------|
| \$3400 | ... | (aliohjelma, joka suorittaa siirron oikealle) |
| \$34FF | RTS | (paluu pääohjelmaan)                          |
| \$3500 | ... | (aliohjelma, joka suorittaa tulituksen)       |
| \$35FF | RTS | (paluu pääohjelmaan)                          |

## Piirrämme kuvaruudulle

Basicissa normaali tapa saada kuvaruudulla jotain aikaan on käyttää PRINT-lauseita. PRINT-lauseen lainausmerkkien sisällä voi olla ruudulle tulostettavia merkkejä, värimäärityksiä ja kursorin eli kohdistimen siirtoja, jotka määräävät, mihin kohtaan kuvaruutua tulostus tapahtuu.

Tämä Commodoren monimutkainen PRINT-mekanismi on vahva ase Basicilla työskenneltäessä. Ohjelman teko on helpompaa ja ohjelmista tulee lyhyempiä ja vastaavasti nopeampia.

Basicilla toimittaessa voimme ohjata kuvaruutua myös suoraan POKE-käskyillä. Kaikki se, mitä kuvaruudulla näkyy, on sijoitettuna muistipaikkoihin \$0400-\$07E7 (1024-2023). Ensimmäiset 40 paikkaa sisältävät ensimmäisen vaakarivin merkit, seuraavat 40 toisen vaakarivin jne. Rivejä on 25, joista koostuu yhteensä 1000 muistipaikkaa. POKE-käskyillä voimme sijoittaa minkä merkin tahansa mihin kohtaan kuvaruutua tahansa.

Kokeile vaikkapa seuraavaa:

**POKE 56295,0**

**POKE 2023,88**

Kuvaruudun oikeaan alakulmaan ilmestyi joko ristimerkki tai X-kirjain, riippuen siitä, oliko kone pienet vai isot kirjaimet -tilassa.

Tämä muistialue ei vielä sisällä merkkien väritietoja. Ne sijaitsevat muistipaikoissa \$D800-\$DBE7 (55296-56295). Muistipaikkojen järjestys on täällä aivan sama kuin merkkimuistissakin, eli ensimmäinen rivi vie 40 ensimmäistä muistipaikkaa jne.

Kokeile nyt seuraavaa:

**POKE 56295,2**

Mikäli POKEamaşi merkki sijaitsee vielä samassa paikassa (ruutu ei ole scrolannut), se muuttui punaiseksi.

Aivan vastaavasti toimimme konekielellä. POKE-käsky vain vaihdetaan konekielen käskypariin LDA ja STA. LDA:lla ladataan ensin akkuun halutun merkin kuvaruutukoodi ja STA:lla se viedään oikeaan kohtaan kuvaruudulla ja vastavasti värikoodi värimuistiin.

Konekielellä et voi suoraan käyttää hyväksesi kursorinohjauksia. Sinun on itse laskettava halutun kuvaruutupaikan osoite ja vietävä koodi sinne STA:lla. Muuttamat koneen valmisrutiineista tosin tarjoavat mahdollisuuden kursoriohjauksen hyväksikäyttöön.

Seuraava ohjelmanosa tulostaa kuvaruudulle A-kirjaimen kursorin sen hetki-  
seen sijaintipaikkaan.

**\$0358 LDA #\$41** (A:n ASCII-koodi on \$41 (65))

**\$035A JSR \$FFD2** (\$:sta kutsutaan CHROUT-rutiinia)

Kursorin sijaintia voidaan muuttaa seuraavalla rutiinilla:

**\$0350 CLC** kun C=0, on kyse kursorin asetuksesta

**\$0351 LDX #\$0A** (X-rekisteriin haluttu rivi)

**\$0353 LDY #\$20** (Y-rekisteriin haluttu sarake)

**\$0355 JSR \$FFF0** (\$FFF0:sta kutsutaan PLOT-rutiinia, joka asettaa  
kursorin (C=0) tai lukee kur-  
sorin aseman X- ja Y-rekistereihin (C=1))

Jos haluat kirjoittaa ruudulle pitempiä viestejä tai muita merkkisarjoja, ei ole  
mielekäästä ladata jokaista erikseen akkuun ja siitä siirtää ruudulle. Käteväntä on  
sijoittaa se ja muut vastaavat merkkijonot kuvaruutukoodien jonoina konekieli-  
sen ohjelman ulkopuolelle, alkuun tai loppuun, ja rakentaa aliohjelma, joka il-  
moitetusta osoitteesta lähtien tulostaa ruudulle merkkijonon halutusta osoitteesta  
alkaan.

Koodijonon ensimmäiseksi tavuksi voi laittaa kyseisen jonon pituuden, jolloin  
aliohjelman ei tarvitse etukäteen tietää edes tulostettavan jonon pituutta. Saman  
asian voi toteuttaa myös sijoittamalla jokaisen koodijonon viimeiseksi luvuksi  
\$00, jonka tulostus aliohjelma sitten tunnistaa ja lopettaa siirtämisen.

Tässä esimerkki mahdollisesta aliohjelmasta, joka siirtää annetusta osoitteesta  
lähtien koodijonon kuvaruudulle. Ennen aliohjelman kutsumista on koodijonon  
alkuosoite sijoitettava osoitteisiin \$FB ja \$FC, vähemmän merkitsevä tavu ensin  
mainittuna. Kohdeosoite kuvaruudulla on sijoitettava vastaavasti paikkoihin  
\$FD ja \$FE. Nämä osoitteet ovat käyttäjälle vapaata tilaa nollasivulla ja soveltu-  
vat erinomaisesti, koska niiden yhteydessä voimme käyttää epäsuoria indeksoitu-  
ja osoitustapoja. Siirrettävän merkkijonon ensimmäinen tavu sisältää siirrettä-  
vien tavujen lukumäärän eli merkkijonon pituuden.

**\$0350 LDY #\$00** (sijoita Y-rekisteriin nolla)

**\$0352 LDA (\$FB),Y** (hae merkkien lukumäärä akkuun)

**\$0354 TAY** (siirrä se Y-rekisteriin)

**\$0355 LDA (\$FB),Y** (hae merkki akkuun, viimeinen merkki ensin. \$FB:ssä ja  
\$FC:ssä oli ennalta asetettu koodijonon alkuosoite)

**\$0357 STA (\$FD),Y** (vie se kuvaruudulle. \$FD:ssä ja \$FE:ssä oli ennalta asetettu  
kuvaruudun osoite)

**\$0359 DEY** (seuraava merkki lopusta alkuunpäin lukien)

**\$035A BNE \$0355** (oliko viimeinen eli onko Y nolla?)

**\$035C RTS** (jos oli, palataan rutiinista)

## COMMODORE 64:N KUVARUUTUKOODIT

| koodi | isot<br>kirjaimet<br>ja grafiikka | pienet ja<br>isot<br>kirjaimet | koodi | isot<br>kirjaimet<br>ja grafiikka                                                   | pienet ja<br>isot<br>kirjaimet                                                       |
|-------|-----------------------------------|--------------------------------|-------|-------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------|
| 0     | @                                 | @                              | 36    | \$                                                                                  | \$                                                                                   |
| 1     | A                                 | a                              | 37    | %                                                                                   | %                                                                                    |
| 2     | B                                 | b                              | 38    | &                                                                                   | &                                                                                    |
| 3     | C                                 | c                              | 39    | ,                                                                                   | ,                                                                                    |
| 4     | D                                 | d                              | 40    | (                                                                                   | (                                                                                    |
| 5     | E                                 | e                              | 41    | )                                                                                   | )                                                                                    |
| 6     | F                                 | f                              | 42    | *                                                                                   | *                                                                                    |
| 7     | G                                 | g                              | 43    | +                                                                                   | +                                                                                    |
| 8     | H                                 | h                              | 44    | ,                                                                                   | ,                                                                                    |
| 9     | I                                 | i                              | 45    | -                                                                                   | -                                                                                    |
| 10    | J                                 | j                              | 46    | .                                                                                   | .                                                                                    |
| 11    | K                                 | k                              | 47    | /                                                                                   | /                                                                                    |
| 12    | L                                 | l                              | 48    | 0                                                                                   | 0                                                                                    |
| 13    | M                                 | m                              | 49    | 1                                                                                   | 1                                                                                    |
| 14    | N                                 | n                              | 50    | 2                                                                                   | 2                                                                                    |
| 15    | O                                 | o                              | 51    | 3                                                                                   | 3                                                                                    |
| 16    | P                                 | p                              | 52    | 4                                                                                   | 4                                                                                    |
| 17    | Q                                 | q                              | 53    | 5                                                                                   | 5                                                                                    |
| 18    | R                                 | r                              | 54    | 6                                                                                   | 6                                                                                    |
| 19    | S                                 | s                              | 55    | 7                                                                                   | 7                                                                                    |
| 20    | T                                 | t                              | 56    | 8                                                                                   | 8                                                                                    |
| 21    | U                                 | u                              | 57    | 9                                                                                   | 9                                                                                    |
| 22    | V                                 | v                              | 58    | :                                                                                   | :                                                                                    |
| 23    | W                                 | w                              | 59    | ;                                                                                   | ;                                                                                    |
| 24    | X                                 | x                              | 60    | <                                                                                   | <                                                                                    |
| 25    | Y                                 | y                              | 61    | =                                                                                   | =                                                                                    |
| 26    | Z                                 | z                              | 62    | >                                                                                   | >                                                                                    |
| 27    | [                                 | [                              | 63    | ?                                                                                   | ?                                                                                    |
| 28    | £                                 | £                              | 64    |  |  |
| 29    | ]                                 | ]                              | 65    |  | A                                                                                    |
| 30    | ^                                 | ^                              | 66    |  | B                                                                                    |
| 31    | <                                 | <                              | 67    |  | C                                                                                    |
| 32    | välilyönti                        | välilyönti                     | 68    |  | D                                                                                    |
| 33    | !                                 | !                              | 69    |  | E                                                                                    |
| 34    | "                                 | "                              | 70    |  | F                                                                                    |
| 35    | #                                 | #                              | 71    |  | G                                                                                    |

| koodi | isot<br>kirjaimet<br>ja grafiikka                                                   | pienet ja<br>isot<br>kirjaimet                                                      | koodi | isot<br>kirjaimet<br>ja grafiikka                                                   | pienet ja<br>isot<br>kirjaimet                                                      |
|-------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|-------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
| 72    |    | H                                                                                   | 101   |    |    |
| 73    |    | I                                                                                   | 102   |    |    |
| 74    |    | J                                                                                   | 103   |    |    |
| 75    |    | K                                                                                   | 104   |    |    |
| 76    |    | L                                                                                   | 105   |    |    |
| 77    |    | M                                                                                   | 106   |    |    |
| 78    |    | N                                                                                   | 107   |    |    |
| 79    |    | O                                                                                   | 108   |    |    |
| 80    |    | P                                                                                   | 109   |    |    |
| 81    |    | Q                                                                                   | 110   |    |    |
| 82    |    | R                                                                                   | 111   |    |    |
| 83    |    | S                                                                                   | 112   |    |    |
| 84    |    | T                                                                                   | 113   |    |    |
| 85    |    | U                                                                                   | 114   |    |    |
| 86    |    | V                                                                                   | 115   |    |    |
| 87    |    | W                                                                                   | 116   |    |    |
| 88    |    | X                                                                                   | 117   |    |    |
| 89    |    | Y                                                                                   | 118   |    |    |
| 90    |    | Z                                                                                   | 119   |    |    |
| 91    |    |    | 120   |    |    |
| 92    |    |    | 121   |    |    |
| 93    |    |    | 122   |    |    |
| 94    |  |  | 123   |  |  |
| 95    |  |  | 124   |  |  |
| 96    | välilyönti                                                                          | välilyönti                                                                          | 125   |  |  |
| 97    |  |  | 126   |  |  |
| 98    |  |  | 127   |  |  |
| 99    |  |  |       |                                                                                     |                                                                                     |
| 100   |  |  |       |                                                                                     |                                                                                     |

POKEamalla tai viemällä STA-käskyllä kuvaruudulle koodiluvun saat ruudulle jomman kumman oikeanpuoleisista merkeistä riippuen siitä, onko koneesi isot vai pienet kirjaimet -tilassa. Luvut 128-255 tuottavat samat merkit kuin luvut 0-127, mutta käänteisinä, eli taustan ja merkin värit vaihdettuna keskenään. Merkin voit siis muuttaa käänteiseksi lisäämällä koodilukuun 128.

## Tarkkuusgrafiikka

Koneiden grafiikkaominaisuuksia arvostellaan yleensä grafiikkatilan pistemäärien perusteella. Commodore 64:n tarkkuusgrafiikassa on 320 x 200 pistettä (kaksivärisenä) tai 160 x 200 (nelivärisenä).

Tarkkuusgrafiikkatilassa jokaisella ruudun pisteellä on muistissa oma bittinsä, joka kertoo onko piste sammutettu vai sytytetty. Tähän tarvitaan yhteensä  $320 \times 200 = 64000$  bittiä eli 8000 tavua. Grafiikkatilaan siirrytään asettamalla muistipaikan \$D011 (53265) 6. bitti eli bitti 5 ykköseksi. Bitithän numeroidaan 0:sta lähtien, joten bitti 5 on järjestysluvultaan kuudes.

Tarkkuusgrafiikan sisältävän muistialueen sijainti voidaan myös määritellä. Ensin grafiikkaruutu asetetaan jonkin 16 kilotavun blokin sisään, koska Commodoren VIC-II -videopiiri pystyy osoittamaan vain 16 kilotavua kerralla. Näitä mahdollisuuksia on neljä: 0-16k, 16-32k, 32-48k ja 48-64k.

Konetta käynnistettäessä VIC-II -piiri asettuu osoittamaan 0-16k blokkia. Blokinvalinta tapahtuu muistipaikan \$DD00 kahden alimman bitin avulla. Konetta käynnistettäessä nämä bitit asettuvat ykkösiksi, eli

- 11 vastaa blokkia 0-16k**
- 10 vastaa blokkia 16-32k**
- 01 vastaa blokkia 32-48k**
- 00 vastaa blokkia 48-64k**

Blokin valinnan jälkeen on 8 kilotavun grafiikkaruutu sijoitettava joko ylemmän tai alemman 8k:n puoliskoon. Se tapahtuu muistipaikan \$D018 (53272) bitin 3 avulla. Kun se on nolla, sijoittuu grafiikkaruutu valitun blokin alemmaan puoliskoon ja vastaavasti ylemmän puoliskoon sen ollessa ykkönen.

Blokin valinnassa on syytä olla varovainen. Oletusarvona käytetty 0-16k ei ole ehkä paras mahdollinen. Alempaa puoliskoa ei voi käyttää, koska se sisältää kaiken koneen toiminnalle tärkeän RAMin: nollasivun, pinomuistin jne. Ylempi puolikas on jo käyttökelpoisempi (osoitteet 8192-16383), koska se sijaitsee Basic-ohjelmille varatulla alueella. Basicilla ohjelmoitaessa on Basic-tilan alkuosoitetta kuitenkin syytä muuttaa tämän alueen yläpuolelle (Basic-tilan alkuosoite sijaitsee muistipaikoissa \$2B ja \$2C, 43 ja 44), jolloin Basic-ohjelmille jää vielä 24 kilotavua tilaa, ennen kuin törmätään Basic-ROMmiin, joka alkaa osoitteesta \$A000 (40960).

Jos VIC-II -piirin muistialuetta siirtää ylöspäin on otettava huomioon, että samalla vaikutetaan kaikkiin tämän piirin osoitettavissa oleviin toimintoihin: merkki-ROM, tekstiruudun sijainti jne.

Jos et hallitse VIC-II -piirin toimintaa riittävän hyvin, on turvallisinta, että koikeilet tarkkuusgrafiikkaa jättämällä blokkivalinnan alkuarvoonsa. Konekieliset ohjelmasihan voit sijoittaa varsin vapaasti, joten siirtämällä grafiikkaruudun blokin yläpuoliskoon olet käyttöjärjestelmän suhteen "kuivilla".

### RAM-parametrit

Tietokoneen käyttöjärjestelmä ja Basic-tulkki pitävät RAM-muistin alapäässä tallessa useita koneen käytön kannalta oleellisia muuttujia, parametreja. Muutamat niistä ovat käyttökelpoisia, kun haluat koneen tekevän jotain omilla ehdoillasi.

Hyvä esimerkki tästä on Basic-ohjelmille varatun tilan loppuosoite (\$37 ja \$38, 55 ja 56), jota muuttamalla saat varattua vapaata tilaa konekielisille ohjelmillesi. Oheisessa muistikartassa on listattuna näistä muistipaikoista hyödyllisimmät ja sellaiset, joiden käyttö ei edellytä monimutkaisten käyttöjärjestelmätoimintojen hallitsemista.

Täydellisemmän luettelon näistä, samaten kuin käyttöjärjestelmän ja Basic-tulkin tarjoamista valmisrutiineista, saat koneesi rakennetta ja toimintaa käsittelevästä hakuteoksesta, joita on lueteltu tämän luvun lopussa.

### Valmisrutiinit

Commodore 64:n KERNAL-ROM ja Basic-ROM tarjoavat monia hyödyllisiä valmisrutiineja, joita normaalisti ohjataan Basic-ohjelmista käsin. Ne ovat kuitenkin kaikki vapaasti myös konekieliohjelmoijan käytössä. On turha keksiä ruutia uudelleen, joten niiden käyttö on myös suositeltavaa ja niiden avulla saat ohjelmasi mahtumaan huomattavasti pienempään tilaan. Ohjelman toimintanopeuteenkaan ne eivät yleensä vaikuta.

Rutiinit käyttävät keskusyksikön rekistereitä hyväkseen. Jos akussa, X- tai Y-rekisterissä on ohjelman kannalta oleellista tietoa, joka ei saa tuhoutua, on ne syytä tallentaa muistiin ennen rutiinin kutsumista. 64:n Programmer's Reference Guide kertoo rutiineista varsin perusteellisesti, myös niiden käyttämistä rekistereistä.

### Kirjallisuutta:

**COMMODORE 64 PROGRAMMER'S REFERENCE GUIDE.**

*VCommodore Business Machines, Inc. 1983*

**KAIKKI KUUSNELOSESTA,** *Tim Onosko, Amersoft 1985*

**COMMODORE 64 OHJELMOINNIN OPAS: KONEKIELI,**

*Anssi Laakkonen ja Mika Huovinen, Amersoft 1985*



## Commodore 64:n RAM-parametreja

| osoite<br>heksana | osoite<br>10-järj. | kuvaus                                                                                          |
|-------------------|--------------------|-------------------------------------------------------------------------------------------------|
| \$0000            | 0                  | osoitteessa 1 sijaitsevan portin tiedon kulkusuunta                                             |
| \$0001            | 1                  | 8-bittinen I/O-portti                                                                           |
| \$0002            | 2                  | <b>KÄYTTÄMÄTÖN TAVU</b>                                                                         |
| \$000A            | 10                 | 0=LOAD, 1=VERIFY. Löytyy myös muistipaikasta 147                                                |
| \$002B-\$002C     | 43- 44             | Basic-ohjelmatilan alkuosoite                                                                   |
| \$0037-\$0038     | 55- 56             | Basic-ohjelmatilan loppuosoite                                                                  |
| \$0073-\$008A     | 115-138            | konek. ohjelma: hae seuraava Basic-tavu (CHRGET)                                                |
| \$0090            | 144                | KERNALin I/O-toimintojen tila. Basicissa varatun sanan ST arvo on sama kuin tämän muistipaikan  |
| \$00A0-\$00A2     | 160-162            | kolmen tavun kellolaskuri, joka kasvaa yhdellä noin 1/60 sekunnin välein                        |
| \$00B2-\$00B3     | 178-179            | kasettipuskurin alkuosoite, kylmäkäynnistyksen jälkeen \$033C (828)                             |
| \$00C5            | 197                | viimeisin näppäinpainallus näppäinkoodina                                                       |
| \$00C6            | 198                | merkkien määrä näppäimistöpuskurissa                                                            |
| \$00C7            | 199                | 0=normaalit merkit, muu arvo=käänteiset merkit                                                  |
| \$00CB            | 203                | voimassa oleva näppäinpainallus näppäinkoodina                                                  |
| \$00CC            | 204                | kursorin vilkutus: 0 = vilkkuu                                                                  |
| \$00CE            | 206                | kursorin alla oleva merkki                                                                      |
| \$00D1-\$00D2     | 209-210            | kursorin kohdalla olevan (loogisen) rivin alkuosoite                                            |
| \$00D3            | 211                | kursorin sijainti loogisella rivillä (0-79)                                                     |
| \$00D6            | 214                | todellinen rivi, jolla kursori sijaitsee (0-24)                                                 |
| \$00FB-\$00FE     | 251-254            | <b>KÄYTTÄMÄTÖN ALUE</b> (4 tavua)                                                               |
| \$0100-\$01FF     | 256-511            | keskusyksikön pinomuisti                                                                        |
| \$0277-\$0280     | 631-640            | näppäimistöpuskuri                                                                              |
| \$0286            | 646                | voimassa oleva merkkien väri                                                                    |
| \$0288            | 648                | kuvaruutumuistin alkusivu (*256=osoite)                                                         |
| \$0291            | 657                | merkistön vaihdon esto: \$00=esto, \$80= sallittu                                               |
| \$02A7-\$02FF     | 679-767            | <b>KÄYTTÄMÄTÖN ALUE</b> (89 tavua)                                                              |
| \$030C            | 780                | keskusyksikön akkurekisteri, SYS-käsky sijoittaa tämän arvon akkuun ennen kutsutun konekielioh- |

|               |           |                                   |
|---------------|-----------|-----------------------------------|
|               |           | jelman käynnistämistä             |
| \$030D        | 781       | keskusyksikön X-rekisteri (SYS)   |
| \$030E        | 782       | keskusyksikön Y-rekisteri (SYS)   |
| \$030F        | 783       | keskusyksikön tilarekisteri (SYS) |
| \$0313        | 784       | KÄYTTÄMÄTÖN TAVU                  |
| \$0334-\$033B | 820-827   | KÄYTTÄMÄTÖN ALUE (8 tavua)        |
| \$033C-\$03FB | 828-1019  | kasettipuskuri (192 tavua)        |
| \$03FC-\$03FF | 1020-1023 | KÄYTTÄMÄTÖN ALUE (4 tavua)        |

Käyttöjärjestelmän varaamat alueet:

|               |             |                                                       |
|---------------|-------------|-------------------------------------------------------|
| \$0400-\$07FF | 1024-2047   | kuvaruudun muistialue                                 |
| \$0800-\$9FFF | 2048-40959  | Basic-ohjelmille varattu tila                         |
| \$A000-\$BFFF | 40960-49151 | Basic-tulkki (ROM) tai 8kt RAM                        |
| \$C000-\$CFFF | 49152-53247 | KÄYTTÄMÄTÖN ALUE (4096 tavua)                         |
| \$D000-\$DFFF | 53248-57343 | I/O-laitteet ja värimuisti tai merkkiROM tai 4 kt RAM |
| \$E000-\$FFFF | 57344-65535 | KERNAL-ROM tai 8 kt RAM                               |

## Hyödyllisiä konekielirutiineja:

Seuraavat sisältyvät KERNALin hyppytaulukkoon ja ne voidaan käynnistää JSR-käskyllä: esimerkiksi JSR \$FFD2. Keskusyksikön rekistereihin on ennen kutsua asetettava mainitut tiedot tai rutiini palauttaa näihin rekistereihin ohjeissa mainitut tiedot.

|        |         |                                                                                                                                                                                                                                                                                                                                                                               |
|--------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$FFCF | (65487) | CHRIN, hae merkki näppäimistöltä (tai muulta avatulta laitteelta), palattaessa merkki on akussa                                                                                                                                                                                                                                                                               |
| \$FFD2 | (65490) | CHROUT, vie merkki kuvaruudulle (tai muulle avatulle laitteelle), merkki asetettava akkuun ennen rutiinin kutsumista                                                                                                                                                                                                                                                          |
| \$FFD5 | (65493) | LOAD, lataa muistiin kasetilta tai levyltä (tai muulta avatulta laitteelta). Ennen rutiinin kutsumista on suoritettava SETLFS ja SETNAM -rutiinit. Jos lataus tapahtuu eri muistialueelle kuin mistä alunperin on tallennettu (SETLFS, Y=0), on X- ja Y-rekistereihin sijoitettava uusi alkuosoite (X=alatu-<br>vu). Akku on asetettava nolaksi (LOAD) tai ykköseksi (VERIFY) |

- \$FFD8 (65496)** SAVE, tallenna muistialue levyille tai kasetille (tai muulle avatulle laitteelle). Ennen rutiinin kutsumista on suoritettava SETLFS ja SETNAM -rutiinit. Tallennus alkaa osoitteesta, joka löytyy nollasivulta akun osoittamasta paikasta ja päättyy X- ja Y-rekistereistä löytyvään osoitteeseen (X=alatuvi)
- \$FFBA (65466)** SETLFS, asettaa loogisen tiedostonumeron (sijoitettava akkuun), laiteosoitteen (sijoitettava X-rekisteriin: 1=kasetinauhuri, 8=levyasema) ja komentotiedostonumeron (sijoitettava Y-rekisteriin) muille KERNAL-rutiineille. Jos komentotiedostonumeroa ei lähetetä, aseta Y-rekisteri 255:ksi.
- \$FFBD (65469)** SETNAM, asettaa tiedostonimen muille rutiineille. Akkuun sijoitetaan ennen kutsua nimen pituus, X- ja Y-rekistereihin nimen alkuosoite muistissa. Jos nimeä ei aseteta, akku=0
- \$FFDB (65499)** SETTIN, ohjelmallisen kellon asetus, kolme tavua, joista eniten merkitsevä sijoitetaan akkuun, loput vastaavasti X- ja Y-rekistereihin
- \$FFDE (65502)** RDTIN, ohjelmallisen kellon luku, tavut palautuvat SETTIN-rutiinia vastaavasti rekistereihin
- \$FFE4 (65508)** GETIN, hae merkki (odottamatta) näppäimistöltä (tai RS-linjalta), palattaessa merkki on akussa
- \$FFED (65517)** SCREEN, lue rivien (Y-rekisteriin) ja sarakkeiden (X-rekisteriin) lukumäärät
- \$FFF0 (65520)** PLOT, lukee (C-lippu=1) kursorin aseman X- ja Y-rekistereihin tai siirtää (C=0) kursorin X- ja Y-rekisterien määräämään paikkaan. X sisältää rivinumeron (0-24) ja Y sarakkenumeron (0-39).



## Konekielinen ohjelmointi

# 9



## VIC-20

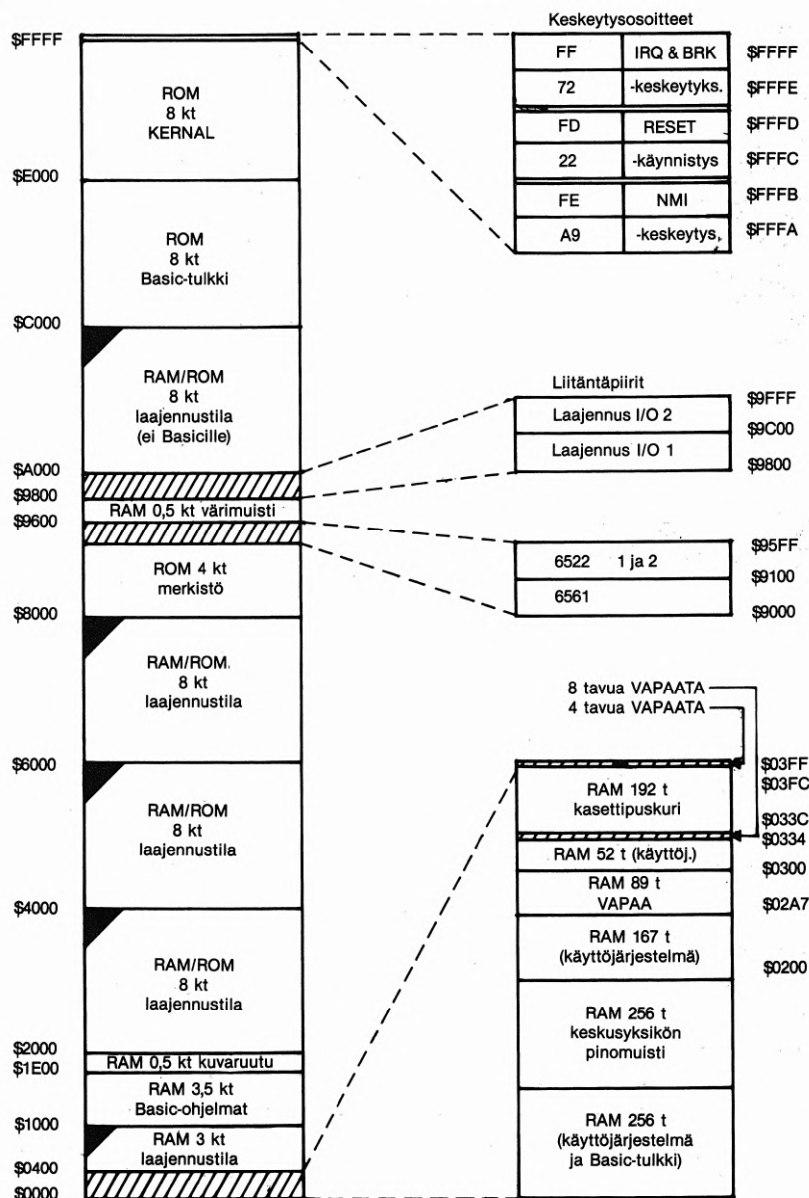
Konekielinen ohjelmointi on harvoin pelkkää numeron pyöritystä muistissa. Toiminnaltaan mielekäs ohjelma joutuu jatkuvasti kommunikoimaan ympäristönsä kanssa. Merkkejä viedään ruudulle, luetaan näppäimistöä ja ohjaimia, joskus nauhuria tai levyasemaa. Käyttöjärjestelmäkin asettaa omalle ohjelmallemme sellaisen ulkoisen tekijän, joka on otettava huomioon.

### Muistikartta

Jotta ulkoisia tekijöitä voi ohjelmissaan käyttää hyväksi on tunnettava niiden sijainti muistialueella ja toimintatapa. Muistikartta on havainnollisin tapa selvittää, mitä tietokoneen sisällä oikein tapahtuu.

Oheisessa kuvassa on VIC-20:n muistikartta, joka ulottuu \$0000:sta \$FFFF:ään eli se sisältää yhteensä 65536 muistipaikkaa. Perusversiossa VICin muistialueesta on käytössä kuitenkin vain osa. Kernal-ROMin (käyttöjärjestelmä) ja Basic-ROMin lisäksi VICin perusversiossa on vain 5,5 kilotavua RAM-muistia, josta Basic-ohjelmille jää 3,5 kilotavua. Kuvaruutu vie 0,5 kilotavua ja värimuisti toiset 0,5 kilotavua. Käyttöjärjestelmä ja Basic-tulkki hallitsevat kilotavun aluetta muistin alapäässä. Pieniä alueita varaavat käyttöönsä laajennus I/O:t ja liitäntäpiirit.

3 kilotavun muistinlaajennusta käytettäessä Basic-ohjelmille varattu tila kasvaa normaalin Basic-tilan alapuolelta 3 kilotavulla. Superexpander käyttää osoitteita



VIC-20:n muistikartta. Perusversiossa VIC sisältää vain 3,5 kilotavua Basic-ohjelmille varattua RAM-muistia. 3 kilotavun lisämuisti ei aiheuta muutoksia muistikarttaan, mutta 8 kilotavua ja sitä suuremmat lisämuistit siirtävät kuva-ruudulle varattua aluetta.

\$6000-\$7FFF ja varaa vielä tilaa muuten vapaalta alueelta 673-767.

Tilanne mutkistuu, kun halutaan suurempia RAM-laajennuksia kuin 3 kilotavuuta. Ne joudutaan sijoittamaan kuvaruudulle varatun alueen yläpuolelle, jolloin Basic-tilasta ei enää tulisi yhtenäinen. Sen vuoksi kuvaruutumuistin paikkaa on siirrettävä osoitteisiin \$1000-\$11FF. Värimuisti siirtyy osoitteisiin \$9400-\$95FF.

Muistia voidaan laajentaa 8 kilotavun paloissa aina 32 kilotavuun saakka. Viimeinen 8 kilotavun pala sijoittuu kuitenkin alueelle, jossa sitä ei voi käyttää Basic-ohjelmien tallennukseen. Siinä voi pyörittää ainoastaan konekieliohjelmia tai Basicin PEEK ja POKE -käskyillä käsiteltäviä tietoja.

## Vapaat alueet

Konekieliohjelmoijalle on ensimmäinen tärkeä asia tietää, missä osoitteissa sijaitsee vapaata tilaa konekieliohjelmien käyttöön. VIC-20:ssä on perusversiossa hyvin vähän vapaata tilaa, jota kone ei käyttäisi lainkaan.

Pisin alue, joka on vapaasti käytettävissä sijaitsee osoitteissa 673-767, joka on sama kuin isoveljessä Commodore 64:ssä. Superexpander tosin käyttää näistä ainakin osoitteita 680, 682-684, 704-706, 712, 715-718, 720 ja 721. Lyhyempiä pätkiä täysin vapaata muistia sijaitsee osoitteissa \$0334-\$033B (820-827) eli 8 tavua ja \$03FC-\$03FF (1020-1023) eli 4 tavua.

Neljän tavun pituiseen tilaan ei mahdu juuri muuta kuin tallennettavia lukuja. Tärkeitä ovat kuitenkin nollasivulla sijaitsevat parin tavun mittaiset vapaat paikat, joita voi käyttää epäsuorien indeksoitujen osoitusmuotojen yhteydessä pitämään sisällään lopullista osoitetta. Kaksi tällaista osoitetta mahtuu muistipaikkoihin \$00FB-\$00FE (251-254).

Nämä olivat siis täysin vapaita paikkoja, joiden käyttäminen ei rajoita koneen muuta toimintaa lainkaan.

Eräs varsin suosittu konekieliohjelmien säilytyspaikka on VICin kasettipuskuri. Se sijaitsee osoitteissa \$033C-\$03FB (828-1019). Se on vapaata tilaa, mutta ainoastaan, jos kasettiasemaa ei käytetä.

Jos haluaa tehdä pitempiä konekielisiä ohjelmia, on RAM-tilaa tavalla tai toisella "lainattava" koneen muilta toiminnoilta. Suurin ja VICin tapauksessa ainoa lainattavissa oleva alue on se tila, jossa käyttäjän omat Basic-ohjelmat ovat tallennettuina. Muistipaikoissa \$0037-\$0038 (55-56) sijaitsee Basicin käytössä olevan muistialueen ylärajan osoite. Käyttäjärjestelmä ja Basic-tulkki käyttävät näiden muistipaikkojen sisältöä hyväkseen sijoittaessaan Basic-ohjelmaa ja sen tuottamia muuttujia muistiin. Voimme käydä muuttamassa tätä osoitetta vaivihkaa ennen kuin Basic-ohjelmaa syötetään koneeseen, jolloin saamme huijattua itsellemme vapaata konekielitilaa. Basicista käsin se käy seuraavalla käskyllä:

**(10)POKE 56,PEEK(56)-4\*X:CLR**

jossa X tarkoittaa tällä tavalla ryöstettyjen kilotavujen määrää. Osoitteessa 56 si-



jaitsee yläpään osoitteen enemmän merkitsevä tavu eli jokainen yksikkö siinä vastaa 256 muistipaikkaa.

Tällä tavalla saamme varattua esimerkiksi 2 kilotavua tilaa käskyllä

### **POKE 56,PEEK(56)-8:CLR**

riippumatta siitä, kuinka paljon VICissämme on lisämuistia. Lisämuistin määrästä on itse laskettava, mistä alue vapautui. 8 kilotavun lisämuistilla se vapautuisi esimerkissämme osoitteesta \$3FFF (16383) alaspäin eli osoitteeseen 14336 asti.

## **Luetaan peliohjainta**

VICissä on yksi peliohjainportti. Se sijaitsee osoitteissa \$911F (37151) ja \$9120 (37152). Muistipaikan 37151 keskimmäiset bitit kuvaavat peliohjaimen kytkimiä seuraavasti:

#### **Muistipaikka 37151**

|                |                   |                           |
|----------------|-------------------|---------------------------|
| <b>bitti 5</b> | <b>tulitus</b>    | <b>PEEK(37151) AND 32</b> |
| <b>bitti 4</b> | <b>vasemmalle</b> | <b>PEEK(37151) AND 16</b> |
| <b>bitti 3</b> | <b>taakse</b>     | <b>PEEK(37151) AND 8</b>  |
| <b>bitti 2</b> | <b>eteen</b>      | <b>PEEK(37151) AND 4</b>  |

#### **Muistipaikka 37152**

|                |                 |                            |
|----------------|-----------------|----------------------------|
| <b>bitti 7</b> | <b>oikealle</b> | <b>PEEK(37152) AND 128</b> |
|----------------|-----------------|----------------------------|

Normaalitilassa bitit ovat ykkösiä. Jos tulitusnäppäintä painetaan, muuttuu 37151:n bitti 5 nollassi, jos sauvaa käännetään vasemmalle, muuttuu bitti 4 nollassi jne. Jos sauvaa käännetään vasemmalle eteen, muuttuvat 37151:n bitit 2 ja 4 nolliksi jne.

Muistipaikan 37151 kanssa voi toimia aivan normaalisti. Bitit voi lukea muistipaikasta ilman harmea, ja Basicissa niitä voi testata esimerkiksi seuraavanlaisella funktiolla: PEEK(37151) AND 4, joka olisi nolla, mikäli peliohjainta olisi käännetty eteenpäin.

Muistipaikan 37152 toimintaan sen sijaan liittyy pieni juju. Samaa muistipaikkaa käytetään myös näppäimistön testaamiseen. Peliohjain ja näppäimistö voivat vaikuttaa toistensa toimintaan, mikäli osaa näppäimistöstä ei ensin kytketä pois päältä sijoittamalla muistipaikkaan 37154 luku 127.

Tämän jälkeen peliohjaimen kääntö oikealle voidaan testata samalla tavalla kuin muutkin asennot funktiolla PEEK(37152) AND 128. Testauksen jälkeen näppäimistö on jälleen kytkettävä päälle sijoittamalla muistipaikkaan 37154 arvo 255. Basicissa se hoidetaan POKE-käskyllä ja konekielessä LDA- ja STA-käskyjen avulla.

Seuraavaa ohjelmanpätäkää voit käyttää omissa ohjelmissasi testaamaan peliohjainten asentoa ja haarautumaan sen mukaan:

|               |            |               |                                                                                                                                        |
|---------------|------------|---------------|----------------------------------------------------------------------------------------------------------------------------------------|
| <b>\$0350</b> | <b>LDA</b> | <b>\$911F</b> | (hae peliohjaimen biti akkuun)                                                                                                         |
| <b>\$0353</b> | <b>LSR</b> |               | (siirrä ensimmäiset kaksi bittiä pois)                                                                                                 |
| <b>\$0354</b> | <b>LSR</b> |               |                                                                                                                                        |
| <b>\$0355</b> | <b>LSR</b> |               | (siirrä C-lippuun ETEEN-bitti)                                                                                                         |
| <b>\$0356</b> | <b>BCS</b> | <b>\$035B</b> | (jos C=1, jatketaan ilman toimenpiteitä)                                                                                               |
| <b>\$0358</b> | <b>JSR</b> | <b>\$0385</b> | (muuten käydään ensin ETEEN-rutiinissa)                                                                                                |
| <b>\$035B</b> | <b>LSR</b> |               | (siirrä C-lippuun TAAKSE-bitti)                                                                                                        |
| <b>\$035C</b> | <b>BCS</b> | <b>\$0361</b> | (jos C=1, jatketaan ilman toimenpiteitä)                                                                                               |
| <b>\$035E</b> | <b>JSR</b> | <b>\$0390</b> | (muuten käydään ensin TAAKSE-rutiinissa)                                                                                               |
| <b>\$0361</b> | <b>LSR</b> |               | (siirrä C-lippuun VASEMMALLE-bitti)                                                                                                    |
| <b>\$0362</b> | <b>BCS</b> | <b>\$0367</b> | (jos C=1, jatketaan ilman toimenpiteitä)                                                                                               |
| <b>\$0364</b> | <b>JSR</b> | <b>\$03A0</b> | (muuten käydään ensin VASEMMALLE-rutiinissa)                                                                                           |
| <b>\$0367</b> | <b>LSR</b> |               | (siirrä C-lippuun TULITUS-bitti)                                                                                                       |
| <b>\$0368</b> | <b>BCS</b> | <b>\$036D</b> | (jos C=1, jatketaan ilman toimenpiteitä)                                                                                               |
| <b>\$036A</b> | <b>JSR</b> | <b>\$03B0</b> | (muuten käydään ensin TULITUS-rutiinissa)                                                                                              |
| <b>\$036D</b> | <b>LDA</b> | <b>#\$7F</b>  | (näppäimistö pois päältä)                                                                                                              |
| <b>\$036F</b> | <b>STA</b> | <b>37154</b>  | (sijoittamalla 37154:ään luku 127 (\$7F))                                                                                              |
| <b>\$0372</b> | <b>LDA</b> | <b>\$9120</b> | (hae peliohjaimen OIKEALLE-bitti akkuun)                                                                                               |
| <b>\$0375</b> | <b>ASL</b> |               | (siirrä OIKEALLE-bitti C-lippuun)                                                                                                      |
| <b>\$0376</b> | <b>LDA</b> | <b>#\$FF</b>  | (näppäimistö jälleen päälle)                                                                                                           |
| <b>\$0378</b> | <b>STA</b> | <b>37154</b>  | (sijoittamalla 37154:ään luku 255(\$FF))                                                                                               |
| <b>\$037B</b> | <b>BCS</b> | <b>\$0380</b> | (jos C=1, jatketaan ilman toimenpiteitä)                                                                                               |
| <b>\$037D</b> | <b>JSR</b> | <b>\$03C0</b> | (muuten käydään ensin OIKEALLE-rutiinissa)                                                                                             |
| <b>\$0380</b> | <b>...</b> |               | (ohjelma jatkuu...)                                                                                                                    |
| <b>\$0385</b> | <b>-</b>   |               | (edellisissä aliohjelmakutsuissa mainitut rutiinit. Rutiinien on tallennettava akku, jotta muutkin suuntabitit voidaan ottaa huomioon. |

## Piirrämme kuvaruudulle

Basicissa normaali tapa saada kuvaruudulla jotain aikaan on käyttää PRINT-lauseita. PRINT-lauseen lainausmerkkien sisällä voi olla ruudulle tulostettavia merkkejä, värimäärytyksiä ja kursorin eli kohdistimen siirtoja, jotka määräävät, mihin kohtaan kuvaruutua tulostus tapahtuu.

Tämä Commodoren monimutkainen PRINT-mekanismi on vahva ase Basicilla työskenneltäessä. Ohjelman teko on helpompaa ja ohjelmista tulee lyhyempiä ja vastaavasti nopeampia.

Basicilla toimittaessa voimme ohjata kuvaruutua myös suoraan POKE-käskyillä. Kaikki se, mitä kuvaruudulla näkyy, on sijoitettuna perusVICissä

muistipaikkoihin \$1E00-\$1FFF (7680-8185). Ensimmäiset 22 paikkaa sisältävät ensimmäisen vaakarivin merkit, seuraavat 22 toisen vaakarivin jne. Rivejä on 23, joista koostuu yhteensä 506 muistipaikkaa. POKE-käskyillä voimme sijoittaa minkä merkin tahansa mihin kohtaan kuvaruutua tahansa.

Kokeile vaikkapa seuraavaa:

**POKE 38905,0:POKE 8185,88**

Kuvaruudun oikeaan alakulmaan ilmestyi joko ristimerkki tai X-kirjain, riippuen siitä, oliko kone pienet vai isot kirjaimet -tilassa.

Tämä muistialue ei vielä sisällä merkkien väritietoja. Ne sijaitsevat muistipaikoissa \$9600-\$97FF (38400-38905). Muistipaikkojen järjestys on täällä aivan sama kuin merkkimuistissakin, eli ensimmäinen rivi vie 22 ensimmäistä muistipaikkaa jne.

Kokeile nyt seuraavaa:

**POKE 38905,2**

Mikäli POKEamasi merkki sijaitsee vielä samassa paikassa (ruutu ei ole scrollannut), se muuttui punaiseksi.

Aivan vastaavasti toimimme konekielellä. POKE-käsky vain vaihdetaan konekielen käskypariin LDA ja STA. LDA:lla ladataan ensin akkuun halutun merkin kuvaruutukoodi ja STA:lla se viedään oikeaan kohtaan kuvaruudulla ja vastavasti värikoodi värimuistiin.

Konekielellä et voi suoraan käyttää hyväksesi kursorinohjauksia. Sinun on itse laskettava halutun kuvaruutupaikan osoite ja vietävä koodi sinne STA:lla. Muutamat koneen valmisrutiineista tosin tarjoavat mahdollisuuden kursoriohjauksen hyväksikäyttöön.

Jos haluat kirjoittaa ruudulle pitempiä viestejä tai muita merkkisarjoja, ei ole mielekäästä ladata jokaista erikseen akkuun ja siitä siirtää ruudulle. Käteväntä on sijoittaa se ja muut vastaavat merkkijonot kuvaruutukoodien jonoina konekielisen ohjelman ulkopuolelle, alkuun tai loppuun, ja rakentaa aliohjelma, joka ilmoitetusta osoitteesta lähtien tulostaa ruudulle merkkijonon halutusta kuvaruutuosoitteesta alkaen. VICin kuvaruutukoodit ovat samat kuin C-64:ssäkin. Löydät ne sivuilta 113-114.

Koodijonon ensimmäiseksi tavuksi voi laittaa kyseisen jonon pituuden, jolloin aliohjelman ei tarvitse etukäteen tietää edes tulostettavan jonon pituutta. Saman asian voi toteuttaa myös sijoittamalla jokaisen koodijonon viimeiseksi luvuksi \$00, jonka tulostusaliohjelma sitten tunnistaa ja lopettaa siirtämisen.

Tässä esimerkki mahdollisesta aliohjelmasta, joka siirtää annetusta osoitteesta lähtien koodijonoa kuvaruudulle. Ennen aliohjelman kutsumista on koodijonon alkuosoite sijoitettava osoitteisiin \$FB ja \$FC, vähemmän merkitsevä tavu ensin mainittuna. Kohdeosoite kuvaruudulla on sijoitettava vastaavasti paikkoihin \$FD ja \$FE. Nämä osoitteet ovat käyttäjälle vapaata tilaa nollasivulla ja soveltuvat

erinomaisesti, koska niiden yhteydessä voimme käyttää epäsuoria indeksoituja osoitustapoja. Siirrettävän merkkijonon ensimmäinen tavu sisältää siirrettävien tavujen lukumäärän eli merkkijonon pituuden.

|               |              |                 |                                                                                                                  |
|---------------|--------------|-----------------|------------------------------------------------------------------------------------------------------------------|
| <b>\$3000</b> | <b>LDY #</b> | <b>\$00</b>     | (sijoita Y-rekisteriin nolla)                                                                                    |
| <b>\$3002</b> | <b>LDA</b>   | <b>(\$FB),Y</b> | (hae merkkien lukumäärä akkuun)                                                                                  |
| <b>\$3004</b> | <b>TAY</b>   |                 | (siirrä se Y-rekisteriin)                                                                                        |
| <b>\$3005</b> | <b>LDA</b>   | <b>(\$FB),Y</b> | (hae merkki akkuun, viimeinen merkki ensin.<br>\$FB:ssä ja \$FC:ssä oli ennalta asetettu koodijonon alku-osoite) |
| <b>\$3007</b> | <b>STA</b>   | <b>(\$FD),Y</b> | (vie se kuvaruudulle. \$FD:ssä ja \$FE:ssä oli ennalta asetettu kuvaruudun osoite)                               |
| <b>\$3009</b> | <b>DEY</b>   |                 | (seuraava merkki lopusta alkuunpäin lukien)                                                                      |
| <b>\$300A</b> | <b>BNE</b>   | <b>\$3005</b>   | (oliko viimeinen eli onko Y nolla?)                                                                              |
| <b>\$300C</b> | <b>RTS</b>   |                 | (jos oli, palataan rutiinista)                                                                                   |

## Haetaan näppäimiltä

Konekieliohjelmienkin on haettava ohjausta käyttäjältään. Peleissä se useimmiten hoituu peliohjaimen välityksellä, mutta muissa tilanteissa luonnollisin yhteydenpitoväline on näppäimistö.

VIC-20:ssä näppäimistöllä on oma pieni puskurinsa. Se tarkoittaa, että näppäimiä painettaessa painetun näppäimen koodi sijoitetaan tietylle muistialueelle ja tietyn rekisterin arvoa kasvatetaan yhdellä. Tämä tapahtuu, vaikkei näppäimistöä luettaisikaan sillä hetkellä. Puskurin pituus on kymmenen merkkiä ja se sijaitsee osoitteissa \$0277-\$0280 (631-640). Puskurissa olevien painallusten lukumäärä voidaan lukea muistipaikasta \$00C6 (198)

Luettaessa näppäimistöä puskurin kautta käydään ensin tarkistamassa, onko siellä näppäilyjä odottamassa. Jos muistipaikan 198 sisältö on nolla, ei tavaraa ole noudettavissa. Jos se on suurempi, voidaan ensimmäinen arvo hakea osoitteesta 631. On muistettava vähentää laskurin arvoa yhdellä (mp 198) ja siirtää puskurin sisältöä.

Jos ohjelmaa kiinnostaa ainoastaan tällä hetkellä painettava näppäin eikä puskuria tarvita, voidaan tieto alhaalla olevasta näppäimestä hakea osoitteesta \$00CB (203). Jos siellä on luku 64, ei mitään näppäintä ole painettu, muuten se sisältää painetun näppäimen koodin.

Kätevimmin näppäinten luku sujuu kuitenkin valmiilla ROM-rutiineilla CHRIN ja GETIN.

## RAM-parametrit

Tietokoneen käyttöjärjestelmä ja Basic-tulkki pitävät RAM-muistin alapäässä tallessa useita koneen käytön kannalta oleellisia muuttujia, parametreja. Muutamat niistä ovat käyttökelpoisia, kun haluat koneen tekevän jotain omilla ehdoilla.

## VICin näppäinkoodit

| näppäin | koodi | näppäin | koodi | näppäin | koodi | näppäin                                                                                | koodi |
|---------|-------|---------|-------|---------|-------|----------------------------------------------------------------------------------------|-------|
| ←       | 8     | A       | 17    | Q       | 48    | Z                                                                                      | 33    |
| 1       | 0     | S       | 41    | W       | 9     | X                                                                                      | 26    |
| 2       | 56    | D       | 18    | E       | 49    | C                                                                                      | 34    |
| 3       | 1     | F       | 42    | R       | 10    | V                                                                                      | 27    |
| 4       | 57    | G       | 19    | T       | 50    | B                                                                                      | 35    |
| 5       | 2     | H       | 43    | Y       | 11    | N                                                                                      | 28    |
| 6       | 58    | J       | 20    | U       | 51    | M                                                                                      | 36    |
| 7       | 3     | K       | 44    | I       | 12    | ,                                                                                      | 29    |
| 8       | 59    | L       | 21    | O       | 52    | .                                                                                      | 37    |
| 9       | 4     | :       | 45    | P       | 13    | /                                                                                      | 30    |
| 0       | 60    | ;       | 22    | @       | 53    | CRSR  | 31    |
| f1      | 39    | +       | 5     | *       | 14    | CRSR<=>                                                                                | 23    |
| f3      | 47    | -       | 61    | ↑       | 54    | INST/DEL                                                                               | 7     |
| f5      | 55    | £       | 6     | =       | 46    | CLR/HOME                                                                               | 62    |
| f7      | 63    | EI PAIN | 64    | RETURN  | 15    | SPACE                                                                                  | 32    |

Hyvä esimerkki tästä on Basic-ohjelmille varatun tilan loppuosoite (\$37 ja \$38, 55 ja 56), jota muuttamalla saat varattua vapaata tilaa konekielisille ohjelmillesi. Oheisessa muistikartassa on listattuna näistä muistipaikoista hyödyllisimmät ja sellaiset, joiden käyttö ei edellytä monimutkaisten käyttöjärjestelmätoimintojen hallitsemista.

Täydellisemmän luettelon näistä, samaten kuin käyttöjärjestelmän ja Basic-tulkin tarjoamista valmisrutiineista, saat koneesi rakennetta ja toimintaa käsittelevästä hakuteoksesta, joita on lueteltu tämän luvun lopussa.

## Valmisrutiinit

VIC-20:n KERNAL-ROM ja Basic-ROM tarjoavat monia hyödyllisiä valmisrutiineja, joita normaalisti ohjataan Basic-ohjelmista käsin. Ne ovat kuitenkin kaikki vapaasti myös konekieliohjelmoijan käytössä. On turha keksiä ruutia uudelleen, joten niiden käyttö on myös suositeltavaa ja niiden avulla saat ohjelmasi mahtumaan huomattavasti pienempään tilaan. Ohjelman toimintanopeuteenkaan ne eivät yleensä vaikuta.

## Kirjallisuutta:

VIC-20 PROGRAMMER'S REFERENCE GUIDE, *Commodore Business Machines, Inc.; Howard Sams. 1982*

MAPPING THE VIC, *Russ Davies, COMPUTE! Publications, Inc. 1984*

## VIC-20:n RAM-parametreja

| osoite<br>heksana | osoite<br>10-järj. | kuvaus                                                                                         |
|-------------------|--------------------|------------------------------------------------------------------------------------------------|
| \$000A            | 10                 | 0=LOAD, 1=VERIFY. Löytyy myös muistipaikasta 147                                               |
| \$002B-\$002C     | 43-44              | Basic-ohjelmatilan alkuosoite                                                                  |
| \$0037-\$0038     | 55-56              | Basic-ohjelmatilan loppuosoite                                                                 |
| \$0073-\$008A     | 115-138            | konek. ohjelma: hae seuraava Basic-tavu (CHRGET)                                               |
| \$0090            | 144                | KERNALin I/O-toimintojen tila. Basicissa varatun sanan ST arvo on sama kuin tämän muistipaikan |
| \$00A0-\$00A2     | 160-162            | kolmen tavun kellolaskuri, joka kasvaa yhdellä noin 1/60 sekunnin välein                       |
| \$00B2-\$00B3     | 178-179            | kasettipuskurin alkuosoite, kylmäkäynnistyksen jälkeen \$033C (828)                            |
| \$00C5            | 197                | viimeisin näppäinpainallus näppäinkoodina                                                      |
| \$00C6            | 198                | merkkien määrä näppäimistöpuskurissa                                                           |
| \$00C7            | 199                | 0=normaalit merkit, muu arvo=käänteiset merkit                                                 |
| \$00CB            | 203                | voimassa oleva näppäinpainallus näppäinkoodina. 64=ei painettu                                 |
| \$00CC            | 204                | kursorin vilkutus: 0 = vilkkuu                                                                 |
| \$00CE            | 206                | kursorin alla oleva merkki                                                                     |
| \$00D1-\$00D2     | 209-210            | kursorin kohdalla olevan (loogisen) rivin alkuosoite                                           |
| \$00D3            | 211                | kursorin sijainti loogisella rivillä (0-79)                                                    |
| \$00D6            | 214                | todellinen rivi, jolla kursori sijaitsee (0-24)                                                |
| \$00FB-\$00FE     | 251-254            | KÄYTTÄMÄTÖN ALUE (4 tavua)                                                                     |
| \$0100-\$01FF     | 256-511            | keskusyksikön pinomuisti                                                                       |
| \$0277-\$0280     | 631-640            | näppäimistöpuskuri                                                                             |
| \$0286            | 646                | voimassa oleva merkkien väri                                                                   |
| \$0288            | 648                | kuvaruutumuistin alkusivu (*256=osoite)                                                        |
| \$0291            | 657                | merkistön vaihdon esto: \$00=esto, \$80= sallittu                                              |
| \$02A1-\$02FF     | 673-767            | KÄYTTÄMÄTÖN ALUE (89 tavua).                                                                   |

|                      |                  |                                                                                                                     |
|----------------------|------------------|---------------------------------------------------------------------------------------------------------------------|
| <b>\$030C</b>        | <b>780</b>       | keskusyksikön akkurekisteri, SYS-käsky sijoittaa tämän arvon akkuun ennen kutsutun konekieliohjelman käynnistämistä |
| <b>\$030D</b>        | <b>781</b>       | keskusyksikön X-rekisteri (SYS)                                                                                     |
| <b>\$030E</b>        | <b>782</b>       | keskusyksikön Y-rekisteri (SYS)                                                                                     |
| <b>\$030F</b>        | <b>783</b>       | keskusyksikön tilarekisteri (SYS)                                                                                   |
| <b>\$0334-\$033B</b> | <b>820-827</b>   | <b>KÄYTTÄMÄTÖN ALUE</b> (8 tavua)                                                                                   |
| <b>\$033C-\$03FB</b> | <b>828-1019</b>  | kasettipuskuri (192 tavua)                                                                                          |
| <b>\$03FC-\$03FF</b> | <b>1020-1023</b> | <b>KÄYTTÄMÄTÖN ALUE</b> (4 tavua)                                                                                   |

**Käyttöjärjestelmän varaamat alueet:**

|                      |                    |                                      |
|----------------------|--------------------|--------------------------------------|
| <b>\$0400-\$0FFF</b> | <b>1024-4095</b>   | 3 kilotavun muistinlaajennus         |
| <b>\$1000-\$1DFF</b> | <b>4096-7679</b>   | 3,5 kilotavun Basic-tila             |
| <b>\$1E00-\$1FFF</b> | <b>7680-8191</b>   | 0,5 kilotavun kuvaruutumuisti        |
| <b>\$2000-\$7FFF</b> | <b>8192-32767</b>  | 3 kertaa 8 kilotavun muistilaajennus |
| <b>\$8000-\$8FFF</b> | <b>32768-36863</b> | merkkiROM                            |
| <b>\$9400-\$95FF</b> | <b>37888-38399</b> | vaihtoehtoinen väriRAM               |
| <b>\$9600-\$97FF</b> | <b>38400-38911</b> | väriRAM                              |
| <b>\$A000-\$BFFF</b> | <b>40960-49151</b> | 8 kilotavun ROM tai RAM laajennus    |
| <b>\$C000-\$DFFF</b> | <b>49152-57343</b> | 8 kilotavun BasicROM                 |
| <b>\$E000-\$FFFF</b> | <b>57344-65535</b> | 8 kilotavun KernalROM                |

## VIC-20:n ROM-rutiineja:

Seuraavat sisältyvät KERNALin hyppytaulukkoon ja ne voidaan käynnistää JSR-käskyllä: esimerkiksi JSR \$FFD2. Keskusyksikön rekistereihin on asetettava ohjeissa mainitut tiedot tai rutiini palauttaa näihin rekistereihin ohjeissa mainitut tiedot.

|        |         |                                                                                                                                                                                                                                                                                                                                                             |
|--------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$FFCF | (65487) | CHRIN, hae merkki näppäimistöltä (tai muulta avatulta laitteelta), palattaessa merkki on akussa                                                                                                                                                                                                                                                             |
| \$FFD2 | (65490) | CHROUT, vie merkki kuvaruudulle (tai muulle avatulle laitteelle), merkki asetettava akkuun ennen rutiinin kutsumista                                                                                                                                                                                                                                        |
| \$FFD5 | (65493) | LOAD, lataa muistiin kasetilta tai levyltä (tai muulta avatulta laitteelta). Ennen rutiinin kutsumista on suoritettava SETLFS ja SETNAM -rutiinit. Jos lataus tapahtuu eri muistialueelle kuin mistä alunperin on tallennettu, on X- ja Y-rekistereihin sijoitettava uusi alkuosoite (X=alatau). Akku on asetettava nol-laksi (LOAD) tai ykköseksi (VERIFY) |
| \$FFD8 | (65496) | SAVE, tallenna muistialue levyille tai kasetille (tai muulle avatulle laitteelle). Ennen rutiinin kutsumis-ta on suoritettava SETLFS ja SETNAM -rutiinit. Tallennus alkaa osoitteesta, joka löytyy nollasivulta akun osoittamasta paikasta ja päättyy X- ja Y-rekistereistä löytyvään osoitteeseen (X=alatau)                                               |
| \$FFBA | (65466) | SETLFS, asettaa loogisen tiedostonumeron (sijoitet-tava akkuun), laiteosoitteen (sijoitettava X-rekisteriin: 1=kasettinauhuri, 8=levyasema) ja ko-mentonumeron (sijoitettava Y-rekisteriin) muille KERNAL-rutiineille. Jos komentonumeroa ei lähete-tä, aseta Y-rekisteri 255:ksi                                                                           |
| \$FFBD | (65469) | SETNAM, asettaa tiedostonimen muille rutiineille. Akkuun sijoitetaan ennen kutsua nimen pituus, X- ja Y-rekistereihin nimen alkuosoite muistissa. Jos ni-meä ei aseteta, akku=0                                                                                                                                                                             |
| \$FFDB | (65499) | SETTIM, ohjelmallisen kellon asetus, kolme tavua, joista eniten merkitsevä sijoitetaan akkuun, loput vastaavasti X- ja Y-rekistereihin                                                                                                                                                                                                                      |
| \$FFDE | (65502) | RDTIM, ohjelmallisen kellon luku, tavut palautuvat                                                                                                                                                                                                                                                                                                          |



## SETTIM-rutiinia vastaavasti rekistereihin

|               |         |                                                                                                                                                                                       |
|---------------|---------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>\$FFE4</b> | (65508) | <b>GETIN</b> , hae merkki (odottamatta) näppäimistöltä (tai RS-linjalta), palattaessa merkki on akussa                                                                                |
| <b>\$FFED</b> | (65517) | <b>SCREEN</b> , lue rivien (Y-rekisteriin) ja sarakkeiden (X-rekisteriin) lukumäärät                                                                                                  |
| <b>\$FFF0</b> | (65520) | <b>PLOT</b> , lukee (C-lippu=1) kursorin aseman X- ja Y-rekistereihin tai siirtää (C=0) kursorin X- ja Y-rekisterien määräämään paikkaan. X sisältää rivi-numeron ja Y sarakenumeron. |

# 10



## APPLE II

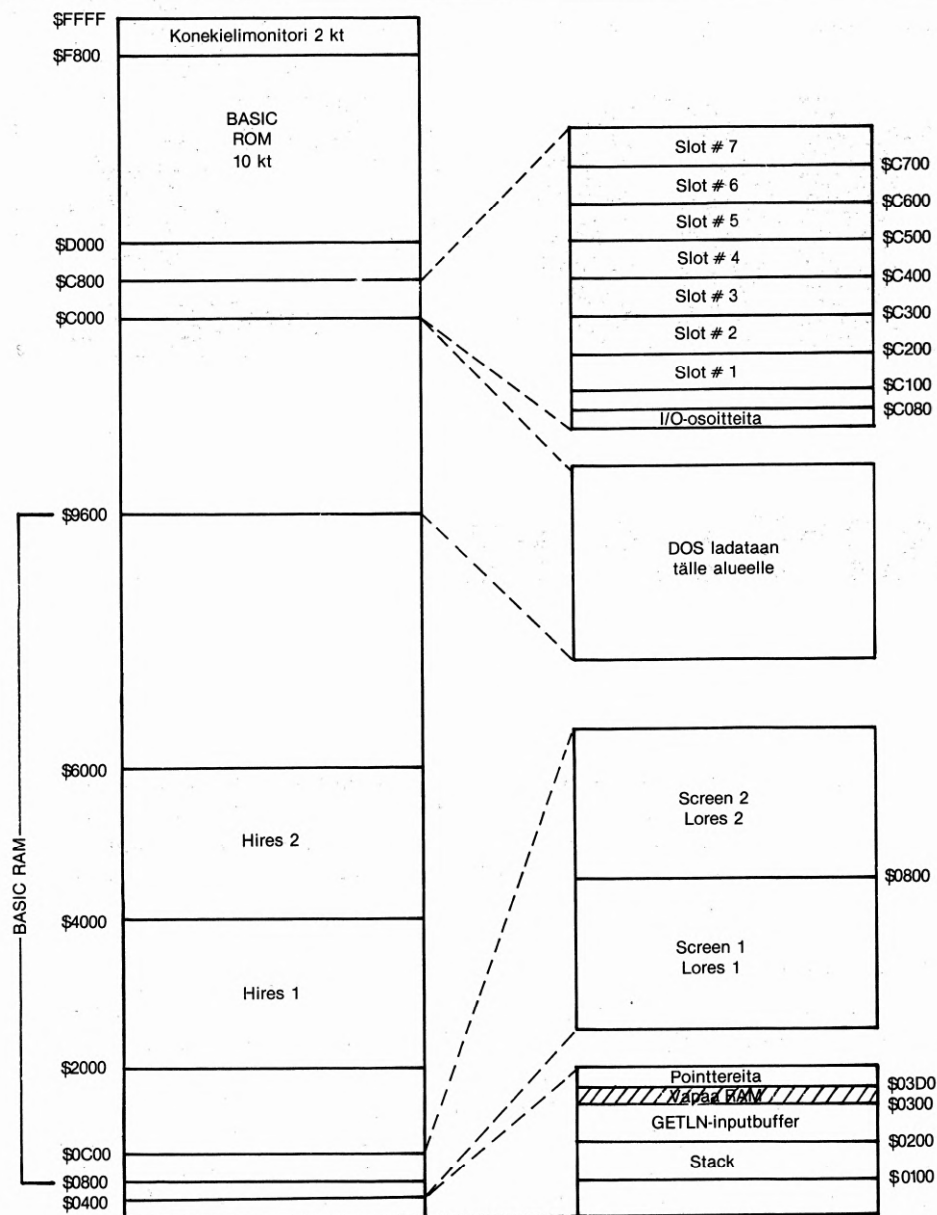
Konekielinen ohjelmointi on harvoin pelkkää numeron pyöritystä muistissa. Toiminnaltaan mielekäs ohjelma joutuu jatkuvasti kommunikoimaan ympäristönsä kanssa. Merkkejä viedään ruudulle, luetaan näppäimistöä ja ohjaimia. Käyttöjärjestelmäkin asettaa omalle ohjelmallemme sellaisen ulkoisen tekijän, joka on otettava huomioon.

### Muistikartta

Jotta ulkoisia tekijöitä voi ohjelmissaan käyttää hyväksi on tunnettava niiden sijainti muistialueella ja toimintatapa. Muistikartta on havainnollisin tapa selvittää, mitä tietokoneen sisällä oikein tapahtuu.

Oheisessa kuvassa on Apple II:n muistikartta, joka ulottuu \$0000:sta \$FFFF:ään eli se sisältää yhteensä 65536 muistipaikkaa. Muistikartta on normaalisti 48 kilotavuisesta Apple II:stä. Muistin alimmat 48 kilotavua ovat RAM (luku/kirjoitus) -muistia ja loput 16 kilotavua ROM (luku) -muistia. ROM muistissa sijaitsee Basic-tulkki ja koneen sisäinen käyttöjärjestelmä. Jos muistiin ladataan vielä levykäyttöjärjestelmä, ryöstää se muistitilaa RAM-alueen yläpäästä.

Muistin sivut 0-7 on varattu koneen omaksi työmuistiksi. Sivulta 8 alkaen sijaitsevat Basic ohjelmat aina levykäyttöjärjestelmään saakka.



Apple II:n, IIe:n ja IIC:n muistikartta. Levykäyttöjärjestelmä ladataan levyllä Basic-alueen yläpäähän. Kuvaruutumuisti ei ole rivi riviltä yhtenäinen, vaan rivit sijaitsevat hieman 'sekaisessa' järjestyksessä.

## Vapaat alueet

Konekieliohjelmoijalle on ensimmäinen tärkeä asia tietää, missä osoitteissa sijaitsee vapaata tilaa konekieliohjelmien käyttöön. Applessa on normaalitilassakin vapaata tilaa, jota kone ei käytä lainkaan. Niihin voi siis tallentaa lyhyitä konekieliohjelmia ilman, että Basiciltakaan viedään lainkaan tilaa. Tämä alue sijaitsee sivulla 3, jolla muistipaikat \$0300-\$03CF eivät ole käyttöjärjestelmän käytössä. Vapaita muistipaikkoja on siten 208 kappaletta.

Jos oma konekieliohjelmasi on pitempi, on sille pakostakin hankittava tilaa muualta, yleensä Basic-ohjelmille varatulta alueelta. Tässä pitää kuitenkin olla varovainen, jotteivat pitkät Basic-ohjelmat tuhoaisi konekieliohjelmaa. Kaikista turvallisin paikka on levykäyttöjärjestelmän (yleensä \$9600-\$BFFF) ja Basic-ohjelmamuistin (yleensä \$0800-\$95FF) välissä.

Jotta pitkät Basic-ohjelmat eivät kirjoittelisi muuttujiaan konekieliohjelman päälle, on Basicin vapaan muistitilan määrää rajoitettava. Tämä saadaan Applella aikaan komennolla HIMEM. Kirjoittamalla HIMEM:xxxx saadaan asetetuksi Basicohjelmamuistin yläraja, josta ylöspäin olevaa aluetta Basic ei käytä (xxxx on tämä muistipaikka, 10-järjestelmän lukuna).

Tätä komentoa käytettäessä tuhoutuu mahdollisesti muistissa oleva Basic-ohjelma. Kannattaa valita Basic-ohjelmamuistin ylärajaksi vaikkapa \$8FFF (des 36863) komennolla HIMEM:36863, jolloin konekielisille ohjelmille jää 1,5 kilotavua tilaa ennen levykäyttöjärjestelmän aluetta. Jos et halua käyttää Basicin HIMEM-komentoa, voit muuttaa osoitteita myös suoraan konekielestä käsin. Jos Basic on Applesoftin valmista (prompti on "A" tai "J"), tallennetaan ylärajan enemmän merkitsevä tavu osoitteeseen \$74 ja vähemmän merkitsevä tavu osoitteeseen \$73. Jos kyseessä on Integer-Basic (prompti on ">"), ovat osoitteet \$4D ja \$4C.

0-sivulta löytyy myös muutamia vapaita paikkoja. Muutaman tavun pituisiin tiloihin ei mahdu juuri muuta kuin tallennettavia lukuja, mutta nollasivulla sijaitsevia parin tavun mittaisia vapaita paikkoja voi käyttää myös epäsuorien indeksoitujen osoitusmuotojen yhteydessä pitämään sisällään lopullista osoitetta. Vapaita 0-sivun paikkoja on mm. \$06-\$09, \$19-\$1F, \$EB-\$EF, \$F9-\$FF.

## Haetaan näppäimiltä

Konekieliohjelmienkin on haettava ohjausta käyttäjältään. Peleissä se useimmiten hoituu peliohjaimen välityksellä, mutta muissa tilanteissa luonnollisin yhteydenpitoväline on näppäimistö.

Näppäimistöltä voi lukea tietoa kahdella tavalla. Joko käytetään koneen käyttöjärjestelmän valmiita komentoja tai luetaan merkit suoraan näppäimistöpiirin

muistipaikoista. Konekielikomennolla **JSR \$FD1B** kone odottaa kunnes näppäimistöltä painetaan näppäintä. Painettu näppäin asettuu kursorin paikalle kuvaruudulle, ja ohjelma palauttaa käyttäjälle merkin koodin akku-rekisterissä.

Näppäimistöä voidaan lukea suoraan muistista muistipaikasta **\$C000**. Bitti 7 ilmaisee näppäimistön tilanteen (1=näppäin painettu, 0=näppäintä ei ole painettu). Samassa osoitteessa on myös annetun merkin koodi, joka vastaa normaaleja merkkikoodeja. Kun merkki on luettu tästä osoitteesta, on vielä nollattava muistipaikka **\$C010**, jossa on toinen näppäimistön tilan ilmoitin.

Aliohjelma, joka lukee näppäimistöltä merkin:

```
$0300 LDA $C000 ;luetaan näppäimistön tila  
$0303 ROL ;siirretään bitti 7 C-lippuun  
$0304 BCC $0300 ;jos C-lippu on 0, luetaan uudestaan  
$0307 ROR ;siirretään bitti 7 C-lipusta takaisin  
$0308 LDX #$00 ; nollataan  
$030A STX $C010 ; muistipaikka $C010  
$030D RTS ;palataan pääohjelmaan
```

## Luetaan peliohjainta

Applessa on yksi ainoa peliohjainportti, johon voi asettaa potentiometrijoystickin tai paddlet. Tavallista viputikkua ei Applessa siis voi käyttää. Peliohjainportin lukua ei kannata yrittää suoraan muistista. Käytön helpottamiseksi on olemassa suora käyttöjärjestelmäaliohjelma osoitteessa **\$FB1E**. Ennen kuin aliohjelma kutsutaan komennolla **JSR \$FB1E** pitää asettaa X-rekisteriin paddlen numero (0 tai 1). Kommentoa voi suoraan verrata Basicin PDL (x) komenttoon. Paddlen arvo ilmestyy Y-rekisteriin.

Peliohjaimen tilan luku lyhyesti:

```
$0300 LDX #$00 ;luetaan paddle nollaa  
$0302 JSR $FB1E ;luetaan tieto, paddlen arvo jää Y-rekisteriin  
$0305 RTS ;palataan käyttäjälle
```

## Kirjoitamme ruudulle

Basicissa normaali tapa saada kuvaruudulla jotain aikaan on käyttää PRINT-lauseita. PRINT-lauseen lainausmerkkien sisällä on ruudulle tulostettavia merkkejä. Kursorin paikkaa ohjataan TAB- ja VTAB -käskyillä.

PRINT-lauseita vastaava käyttöjärjestelmäalirutiini sijaitsee osoitteessa **\$FDF0**. Kirjoitettavan merkin koodin pitää olla akku-rekisterissä. Kursorin paikan saa asetettua käyttöjärjestelmän aliohjelmalla osoitteessa **\$FC24**. Ennen rutiinin kutsumista on asetettava kursorin sijainnin X-arvo (sarake) osoitteeseen **\$24** ja Y-arvo (rivi) osoitteeseen **\$25**. Ruudun saa tyhjennettyä helposti komennolla **JSR \$FC58**,

joka tekee saman kuin Basicin HOME-komento. Applessa ei kannata yrittää suoraan kuvaruutumuistiin kohdistuvaa merkkipokeilua.

Tässä esimerkkiohjelma, joka asettaa kursorin haluttuun paikkaan ja tulostaa tekstirivin. Se saa olla enintään 256 merkkiä pitkä. Tekstin loppuun on asetettava arvo \$00, jotta ohjelma ymmärtäisi tulostettavan kirjoituksen loppuvan siihen. Teksti saa olla missä tahansa muistissa. Sen alkuosoite on asetettava osoitteisiin \$FE (vähemmän merkitsevä tavu) ja \$FF (enemmän merkitsevä tavu). Tekstin tuleva alkupaikka ruudulla on annettava osoitteisiin \$FC (rivi) ja \$FD (sarake). Tämä alirutiini voisi sijaita vaikkapa muistiosoitteesta \$0300 alkaen.

#### Esimerkkiohjelma:

|        |     |          |                                       |
|--------|-----|----------|---------------------------------------|
| \$0300 | LDA | \$FC     | ;luetaan kirjoituksen kohteen x-arvo  |
| \$0302 | LDX | \$FD     | ;luetaan kirjoituksen kohteen y-arvo  |
| \$0304 | STA | \$24     |                                       |
| \$0306 | STX | \$25     |                                       |
| \$0308 | JSR | \$FC24   | ; asetetaan kursori                   |
| \$030B | LDY | #\$00    | ;Y-rekisteri on tekstin kohdistin     |
| \$030D | LDA | (\$FE),Y | ;luetaan Y:des merkki akkuun          |
| \$030F | BEQ | \$031C   | ;jos merkin koodi on \$00, lopetetaan |
| \$0311 | STY | \$FD     | ;tallennetaan Y-rekisteri             |
| \$0313 | JSR | \$FDF0   | ;tulostetaan akussa oleva merkki      |
| \$0316 | LDY | \$FD     | ;luetaan kohdistin Y-rekisteriin      |
| \$0318 | INY |          | ;siirretään kohdistin yksi eteenpäin  |
| \$0319 | JMP | \$030D   | ;hypätään silmukan alkuun             |
| \$031C | RTS |          | ;palataan pääohjelmaan                |

## Apple II:n kuvaruutukoodit

| Koodi | Tulostuva merkki | Koodi | Tulostuva merkki | Koodi | Tulostuva merkki | Koodi | Tulostuva merkki |
|-------|------------------|-------|------------------|-------|------------------|-------|------------------|
| 0     | @                | 32    | välilyönti       | 64    | @                | 96    | välilyönti       |
| 1     | A                | 33    | !                | 65    | A                | 97    | !                |
| 2     | B                | 34    | "                | 66    | B                | 98    | "                |
| 3     | C                | 35    | #                | 67    | C                | 99    | #                |
| 4     | D                | 36    | \$               | 68    | D                | 100   | \$               |
| 5     | E                | 37    | %                | 69    | E                | 101   | %                |
| 6     | F                | 38    | &                | 70    | F                | 102   | &                |
| 7     | G                | 39    | '                | 71    | G                | 103   | '                |
| 8     | H                | 40    | (                | 72    | H                | 104   | (                |
| 9     | I                | 41    | )                | 73    | I                | 105   | )                |
| 10    | J                | 42    | *                | 74    | J                | 106   | *                |
| 11    | K                | 43    | +                | 75    | K                | 107   | +                |
| 12    | L                | 44    | ,                | 76    | L                | 108   | ,                |
| 13    | M                | 45    | -                | 77    | M                | 109   | -                |
| 14    | N                | 46    | .                | 78    | N                | 110   | .                |
| 15    | O                | 47    | /                | 79    | O                | 111   | /                |
| 16    | P                | 48    | 0                | 80    | P                | 112   | 0                |
| 17    | Q                | 49    | 1                | 81    | Q                | 113   | 1                |
| 18    | R                | 50    | 2                | 82    | R                | 114   | 2                |
| 19    | S                | 51    | 3                | 83    | S                | 115   | 3                |
| 20    | T                | 52    | 4                | 84    | T                | 116   | 4                |
| 21    | U                | 53    | 5                | 85    | U                | 117   | 5                |
| 22    | V                | 54    | 6                | 86    | V                | 118   | 6                |
| 23    | W                | 55    | 7                | 87    | W                | 119   | 7                |
| 24    | X                | 56    | 8                | 88    | X                | 120   | 8                |
| 25    | Y                | 57    | 9                | 89    | Y                | 121   | 9                |
| 26    | Z                | 58    | :                | 90    | Z                | 122   | :                |
| 27    | [                | 59    | ;                | 91    | [                | 123   | ;                |
| 28    | \                | 60    | <                | 92    | \                | 124   | <                |
| 29    | ]                | 61    | =                | 93    | ]                | 125   | =                |
| 30    | ^                | 62    | >                | 94    | ^                | 126   | >                |
| 31    | -                | 63    | ?                | 95    | -                | 127   | ?                |

*Apple II:n merkkikoodit. Merkit tulostuvat sellaisenaan ruudulle, lihavalla painetut merkit käänteisinä, kursiivilla painetut välkkyvinä.*

| Koodi | Tulostuva merkki | Koodi | Tulostuva merkki | Koodi | Tulostuva merkki | Koodi | Tulostuva merkki |
|-------|------------------|-------|------------------|-------|------------------|-------|------------------|
| 128   | @                | 160   | välilyönti       | 192   | @                | 224   | ,                |
| 129   | A                | 161   | !                | 193   | A                | 225   | a                |
| 130   | B                | 162   | "                | 194   | B                | 226   | b                |
| 131   | C                | 163   | #                | 195   | C                | 227   | c                |
| 132   | D                | 164   | \$               | 196   | D                | 228   | d                |
| 133   | E                | 165   | %                | 197   | E                | 229   | e                |
| 134   | F                | 166   | &                | 198   | F                | 230   | f                |
| 135   | G                | 167   | '                | 199   | G                | 231   | g                |
| 136   | H                | 168   | (                | 200   | H                | 232   | h                |
| 137   | I                | 169   | )                | 201   | I                | 233   | i                |
| 138   | J                | 170   | *                | 202   | J                | 234   | j                |
| 139   | K                | 171   | +                | 203   | K                | 235   | k                |
| 140   | L                | 172   | ,                | 204   | L                | 236   | l                |
| 141   | M                | 173   | -                | 205   | M                | 237   | m                |
| 142   | N                | 174   | .                | 206   | N                | 238   | n                |
| 143   | O                | 175   | /                | 207   | O                | 239   | o                |
| 144   | P                | 176   | 0                | 208   | P                | 240   | p                |
| 145   | Q                | 177   | 1                | 209   | Q                | 241   | q                |
| 146   | R                | 178   | 2                | 210   | R                | 242   | r                |
| 147   | S                | 179   | 3                | 211   | S                | 243   | s                |
| 148   | T                | 180   | 4                | 212   | T                | 244   | t                |
| 149   | U                | 181   | 5                | 213   | U                | 245   | u                |
| 150   | V                | 182   | 6                | 214   | V                | 246   | v                |
| 151   | W                | 183   | 7                | 215   | W                | 247   | w                |
| 152   | X                | 184   | 8                | 216   | X                | 248   | x                |
| 153   | Y                | 185   | 9                | 217   | Y                | 249   | y                |
| 154   | Z                | 186   | :                | 218   | Z                | 250   | z                |
| 155   | [                | 187   | ;                | 219   | [                | 251   | {                |
| 156   | \                | 188   | <                | 220   | \                | 252   |                  |
| 157   | ]                | 189   | =                | 221   | ]                | 253   | }                |
| 158   | ^                | 190   | >                | 222   | ^                | 254   | ~                |
| 159   | _                | 191   | ?                | 223   | -                | 255   | DEL              |



## Tarkkuusgrafiikka

Koneiden grafiikkaominaisuuksia arvostellaan yleensä grafiikkatilan pistemäärien perusteella. Applen tarkkuusgrafiikassa on normaalisti 256 x 192 pistettä (Hires, nelivärisenä) tai 48 x 40 pistettä (Lores, 16 väriä).

Grafiikka saadaan päälle niinkin helposti kuin lukemalla vain kerran osoitetta \$C050. Tekstiruutu saadaan takaisin lukemalla osoitetta \$C051. Jos halutaan grafiikan ohella ruudun alareunaan neljä tekstiriviä, luetaan osoitetta \$C053 kerran (pois päältä osoitteesta \$C052). Grafiikkatila valitaan osoitteista \$C056 ja \$C057. Lukemalla osoitetta \$C056, valitaan Lores-grafiikka. Hires valitaan lukemalla osoitetta \$C057.

Ensimmäisen Hires-ruudun muistialue on \$2000-\$3FFF ja toisen \$4000-\$5FFF. Ensimmäisen Lores ruudun muistialue on \$0400-\$07FF ja toisen \$0800-\$0BFF. Lores-ruutuun voi kirjoittaa suoraan käyttöjärjestelmälirutiineilla. Ikävä kyllä HiRes-ruudun grafiikkatieto on koodattu mahdollisimman epämukavalla tavalla, joten sen selostaminen ei tässä ole mahdollista.

## Merkistö

Applen käyttämä merkistö on pääosiltaan ASCII-standardin mukainen. Itse merkit jaetaan neljään ryhmään, joihin jokaiseen kuuluu 32 merkkiä. Karkea jako on ohjausmerkit, erikoismerkit, pienet kirjaimet, isot kirjaimet.

Ohjausmerkit (koodit 0-31) eivät näy kuvaruudulla. Niiden ainoa tarkoitus on tarjota mahdollisuus lukea erikoistietoja, kuten esimerkiksi kursorin siirto. Koodit 32-64 sisältävät kaikki erikoismerkit. Erikoismerkkejä ovat kaikki ei-aakkoselliset merkit. Tähän ryhmään lasketaan välilyönti, lainausmerkki, numerot, jne. Isot kirjaimet löytyvät koodeista 65-96, pienet 97-127. Koodit 128-155 sisältävät samat merkit kuin koodit 0-127, mutta käänteisinä.

## RAM-parametrit

Tietokoneen käyttöjärjestelmä ja Basic-tulkki pitävät RAM-muistin alapäässä tallessa useita koneen käytön kannalta oleellisia muuttujia, parametreja. Muutamat niistä ovat käyttökelpoisia, kun haluat koneen tekevän jotain omilla ehdoillasi.

Hyvä esimerkki tästä on Basic-ohjelmille varatun tilan loppuosoite (Integer Basic: \$4C ja \$4D, Applesoft-Basic: \$73, \$74), jota muuttamalla saat varattua vapaita tilaa konekielisille ohjelmillesi. Oheisessa muistikartassa on listattuna näistä muistipaikoista hyödyllisimmät ja sellaiset, joiden käyttö ei edellytä monimutkaisten käyttöjärjestelmätoimintojen hallitsemista.

Täydellisemmän luettelon näistä, samaten kuin käyttöjärjestelmän ja Basic-tulkin tarjoamista valmisrutiineista, saat koneesi rakennetta ja toimintaa käsittelevästä hakuteoksesta, joita on lueteltu tämän luvun lopussa.

## Valmisrutiinit

Applen käyttöjärjestelmä tarjoaa monia hyödyllisiä valmisrutiineja, joita normaalisti ohjataan Basic-ohjelmista käsin. Ne ovat kuitenkin kaikki vapaasti myös konekieliohjelmoijan käytössä. On turha keksiä ruutia uudelleen, joten niiden käyttö on myös suositeltavaa ja niiden avulla saat ohjelmasi mahtumaan huomattavasti pienempään tilaan. Ohjelman toimintanopeuteenkaan ne eivät yleensä vaikuta.

## Kirjallisuutta:

SAMS APPLE PROGRAMMER'S HANDBOOK, *Paul Irwin*

APPLE II REFERENCE GUIDE (Jokaiselle konemallille oma kirja)

## Applen RAM-parametrit

| Osoite        | Selitys                                                                                                                                |
|---------------|----------------------------------------------------------------------------------------------------------------------------------------|
| \$0030        | Lores-piirtoväri. Sisältö = piirtoväri * \$11                                                                                          |
| \$0033        | GETLN-funktion prompti (merkkikoodi)                                                                                                   |
| \$004A-004B   | Integer Basicin alin Basic-ohjelmalle vapaa tavu muodossa LO+HI-byte, asetetaan Basicista käsin LOMEM-komennolla. Normaali-arvo \$0801 |
| \$004C-004D   | Integer Basicin muistitilan yläraja HIMEM                                                                                              |
| \$0067-0068   | Applesoft Basicin LOMEM                                                                                                                |
| \$0073-0074   | Applesoft Basicin ensimmäinen ei-vapaa tavu, HIMEM+1                                                                                   |
| \$0200-\$02FF | GETLN-funktiolla sisäänsyötetty input-rivi, syötetty teksti loppuu RETURNin koodiin (merkkikoodi \$8D)                                 |
| \$0300-03CF   | Vapaata muistilaa omille konekieliohjelmille                                                                                           |
| \$03F0-03F1   | BRK-komennon käsittelevän ohjelman hyppyosoite, normaalisti \$59 ja \$FA                                                               |
| \$0400-07FF   | Kuvaruutumuisti sekä ensimmäinen Lores-grafiikkaruutu                                                                                  |
| \$0800-0BFF   | Toinen Lores-ruutu                                                                                                                     |
| \$0800-95FF   | Normaalisti vapaa muistialue Basicille (ja konekieliohjelmille)                                                                        |
| \$2000-3FFF   | Ensimmäinen Hires-grafiikkaruutu                                                                                                       |
| \$4000-5FFF   | Toinen Hires-ruutu                                                                                                                     |
| \$C000        | Sisältää painetun näppäimen koodin                                                                                                     |
| \$C050        | Kuvaruutu grafiikkamuotoon (*)                                                                                                         |
| \$C051        | Kuvaruutu tekstimuotoon (*)                                                                                                            |
| \$C052        | Grafiikkaruutu käsittää koko ruudun (*)                                                                                                |
| \$C053        | Grafiikkaruudun alareunaan 4 tekstirivä (*)                                                                                            |
| \$C054        | Valitsee ensimmäisen Hires/Lores-ruudun (*) (**)                                                                                       |
| \$C055        | Valitsee toisen Hires/Lores-ruudun (*) (**)                                                                                            |
| \$C056        | Grafiikkaruutu Lores-muotoon (*)                                                                                                       |
| \$C057        | Grafiikkaruutu Hires-muotoon (*)                                                                                                       |
| \$C061        | Ohjainkytkin 0, bitti 7 kertoo tilanteen (1=painettu), vastaa OPEN-APPLE-näppäintä                                                     |
| \$C062        | Ohjainkytkin 1, bitti 7 kertoo tilanteen, vastaa CLOSED-APPLE-näppäintä                                                                |
|               | (*) Osoitetta lukemalla saadaan aikaan selostettu toiminto                                                                             |
| 142           | (**) Jos käytetään 80-merkkistä ruutua, toiminnot muuttuvat                                                                            |

## Applen ROM-rutiinit

Koneen ROMissa olevia valmiita rutiineja, jotka käynnistetään käskyllä JSR x (x=toiminnon osoite). Ennen kutsua on akkuun, X- ja Y- rekistereihin sijoitettava mainitut tiedot.

| Osoite      | Toiminto                                                                                                                                                                                                                           |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$C100-C700 | aktivoidaan lisäkortteja käskyllä JSR \$Cx00, jolloin x on korttipaikan numero. Vastaa Basicin PR#-komentoa                                                                                                                        |
| \$E5E2      | AppleSoft Basicin muistinsiirtoalirutiini alle 255 merkkiä pitkille muistinpätkille. Siirrettävän alueen alku Y- (high) ja X- (low) rekistereihin, pituus akku-rekisteriin. Kohdeosoite muistipaikkoihin \$71 (low) ja \$72 (high) |
| \$F800      | PLOTataan ensimmäiselle Lores-ruudulle. Piste x-koordinaatti akku-rekisteriin ja y-koordinaatti Y-rekisteriin.                                                                                                                     |
| \$F819      | Piirtää horisontaalisen viivan ensimmäiselle Lores-ruudulle. Alkupisteen x-koordinaatti Y-rekisteriin, y-koordinaatti akku-rekisteriin, sekä loppupisteen x-koordinaatti muistipaikkaan \$2C.                                      |
| \$F828      | Piirtää pystysuoran viivan ensimmäiselle Lores-ruudulle. Alkupisteen x-koordinaatti Y-rekisteriin ja y-koordinaatti akku-rekisteriin. Loppupisteen y-koordinaatti osoitteeseen \$2D.                                               |
| \$F832      | Tyhjentää koko ensimmäisen Lores-ruudun.                                                                                                                                                                                           |
| \$F836      | Tyhjentää Lores-ruudun tuhoamatta ruudun alimpia neljää riviä (jos niillä on tekstiä).                                                                                                                                             |
| \$F864      | Asettaa Lores-piirtoväriin, ennen kutsua täytyy asettaa akku-rekisteriin väriin numero (0-15).                                                                                                                                     |
| \$F871      | Haetaan määrätyn pisteen väri. x-koordinaatti Y-rekisteriin, y-koordinaatti akku-rekisteriin. Tulos tulee akku-rekisteriin.                                                                                                        |
| \$F941      | Tulostaa neljämerkkisen heksaluvun, akku-rekisteriin Hi-byte ja X-rekisteriin Low-byte.                                                                                                                                            |
| \$FB1E      | Vastaa Basicin PDL (x) komentoa, paddlen numero X-rekisteriin, tulos tulee Y-rekisteriin.                                                                                                                                          |
| \$FB2F      | Vastaa Basicin komentoa TEXT.                                                                                                                                                                                                      |
| \$FB40      | Vastaa Basicin komentoa GR.                                                                                                                                                                                                        |
| \$FBDD      | Tekee beep-äänen, jonka taajuus on 1000 Hz.                                                                                                                                                                                        |
| \$FC24      | Asettaa kursorin, tuleva x-koordinaatti muistipaikkaan \$24, y-koordinaatti muistipaikkaan \$25.                                                                                                                                   |

|        |                                                                                                                                                                                                 |
|--------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$FC58 | Vastaa Basicin HOME-käskyä.                                                                                                                                                                     |
| \$FD1B | Odottaa yhtä merkkiä näppäimistöltä, merkki tulee akku-rekisteriin.                                                                                                                             |
| \$FD6A | GETLN-rutiini, vastaa Basicin INPUT-komentoa. Sisäänsyötetty rivi tulee muistialueelle \$0200-02FF alkaen osoitteesta \$0200. Rivi loppuu muistissa CR:ään (merkkikoodi \$8D)                   |
| \$FDDA | Tulostaa kaksimerkkisen heksaluvun, tulostettava numero oltava akkurekisterissä.                                                                                                                |
| \$FDF0 | Tulostetaan yksi merkki ruudulle, merkin koodi on oltava akku-rekisterissä.                                                                                                                     |
| \$FE2C | Muistialueiden siirtorutiini. Siirrettävän alueen alkuosoite muistipaikkoihin \$3C (low) ja \$3D (high), loppusointe muistipaikkoihin \$3E ja \$3F. Kohdealueen alku osoitteisiin \$42 ja \$43. |
| \$FF69 | Konekielimonitorin aloituskohta johon hyppäämällä pääsee konekielimonitoriin. (Basicista käsin komennolla CALL -151)                                                                            |

Konekielinen ohjelmointi

# 11



## ATARI 600XL, 800XL JA 130XE

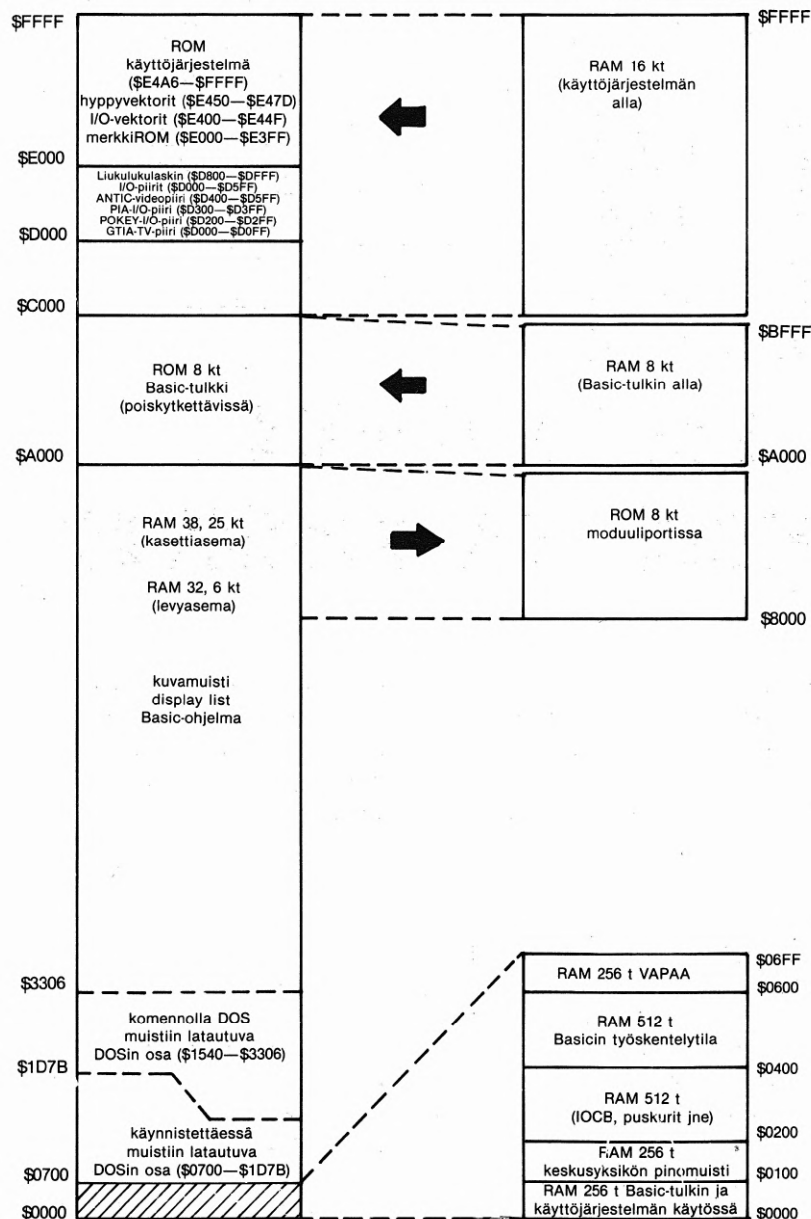
Konekielinen ohjelmointi on harvoin pelkkää numeron pyöritystä muistissa. Toiminnaltaan mielekäs ohjelma joutuu jatkuvasti kommunikoimaan ympäristönsä kanssa. Merkkejä viedään ruudulle, luetaan näppäimistöä ja ohjaimia, joskus nauhuria tai levyasemaa. Käyttöjärjestelmäkin asettaa omalle ohjelmalle sellaisen ulkoisen tekijän, joka on otettava huomioon.

### Muistikartta

Jotta ulkoisia tekijöitä voi ohjelmissaan käyttää hyväksi on tunnettava niiden sijainti muistialueella ja toimintatapa. Muistikartta on havainnollisin tapa selvittää, mitä tietokoneen sisällä oikein tapahtuu.

Oheisessa kuvassa on Atarin muistikartta, joka ulottuu \$0000:sta \$FFFF:ään eli se sisältää yhteensä 65536 muistipaikkaa. Erikoisin ominaisuus Atarin muistissa on sen kaksijakoisuus: koko 64 kilotavun alue on RAM-muistin (RAM=luku/kirjoitusmuisti) peitossa, mutta siitä huolimatta muistialueella sijaitsevat mm. käyttöjärjestelmä ja Basic-tulkki, jotka ovat ROM-muistia (lukumuistia). Normaaliutilassa keskusyksikkö kommunikoi näiden ROM-muistien ja niiden ulkopuolelle jäävien RAM-muistialueiden kanssa. Osa RAM-muistista jää siis piiloon ROMin alle. Nämä RAM-muistialueet saadaan käyttöön eli pystytään lukemaan, jos muistipaikan 54017 (\$D301) kahden ensimmäisen bitin arvoa muutetaan.

Lisäksi yksi 2kt:n ROM-lohko, nimittäin itsetesti jää RAM-muistin alle. Vaih-



**Atarin muistikartta. 600XL:n 16 kt:n RAM-muisti päättyy muistipaikkaan \$3FFF (16383). 130XE:hen lisättyyn 64 kt muistiin pääsee käsiksi 16 kt:n lohkoissa muistipaikan SD301 (54017) bittien 3—6 avulla.**

tamalla muistipaikan \$D301 seitsemännen bitin arvoa pystytään lukemaan myös itsetestin sisältävää ROM-aluetta 20480-22527.

Omassa konekieliohjelmassaan voi Basic-tulkin huoletta kytkeä pois päältä, mikäli ohjelma ei käytä Basicin alirutiineja. Käyttöjärjestelmän pois kytkeminen onkin jo mutkikkaampi asia. Käyttöjärjestelmä pitää nimittäin huolen keskusyksikön keskeytyksistä ja keskeytysrutiiniosoitteista. Käyttöjärjestelmää pois kytkettäessä on varmistettava näiden toimintojen jatkuminen.

## Vapaat alueet

Konekieliohjelmoijalle on ensimmäinen tärkeä asia tietää, missä osoitteissa sijaitsee vapaata tilaa konekieliohjelmien käyttöön. Atarissa on normaalitilassakin useita kilotavuja vapaata tilaa, jota kone ei käytä lainkaan. Niihin voi siis tallentaa omia konekieliohjelmiaan ilman, että Basiciltakaan viedään lainkaan tilaa.

Tärkein näistä alueista on "sivu kuusi" eli 256 tavun pituinen käyttäen konekieliohjelmitte tarkoitettu alue \$600-\$6FF (1536-1791). Tästä alueesta skandinaavisilla näppäimillä varustettu 800XLS käyttää viimeisen muistipaikan (1791).

Lyhyempi vapaa alue sijaitsee osoitteissa \$CB-\$D1 (203-209) eli 7 tavua. Seitsemän tavun pituiseen tilaan ei mahdu juuri muuta kuin tallennettavia lukuja. Nollasivulla sijaitsevat muutaman tavun mittaiset vapaat paikat ovat kuitenkin tärkeitä, koska niitä voi käyttää epäsuorien indeksoitujen osoitusmuotojen yhteydessä pitämään sisällään alkuperäistä osoitetta.

130 XE:ssä on koko toinen 64 kt:n muistilohko täysin vapaa. Vain harvat ohjelmat hyödyntävät tätä aluetta, käytännössä vain DOS 2.5.

Nämä olivat siis täysin vapaita paikkoja, joiden käyttäminen ei rajoita koneen muuta toimintaa lainkaan.

Varsin suosittuja konekieliohjelmien säilytyspaikkoja ovat kasettipuskuri ja printteripuskuri. 128 tavun pituinen kasettipuskuri sijaitsee osoitteissa \$3FD-\$47F (1021-1151) ja 40 tavun pituinen printteripuskurin muistipaikoissa \$3C0-\$3E7 (960-999). Ne ovat vapaata tilaa, mutta ainoastaan, jos kasettiasemaa tai kirjoitinta ei käytetä.

Basic-ohjelman yhteydessä toimivat konekieliohjelmat voi tallentaa myös merkkijonoihin, kuten KONEK\$, jolloin ne käynnistetään käskyllä:

**USR(ADR(KONEK\$))**

Ohjelmat sijaitsevat tällöin merkkijonomuuttujille varatulla alueella ja ovat täysin suojassa Basicilta. Ohjelmien on kuitenkin oltava vapaastisijoitettavia.

Jos haluaa tehdä pitempiä konekielisiä ohjelmia, on RAM-tilaa tavalla tai toisella "lainattava" koneen muilta toiminnoilta. Suurin lainattavissa oleva alue on se 38 kilotavun tila, missä käyttäjän omat Basic-ohjelmat ovat tallennettuina. Muistipaikassa \$006A (106) sijaitsee Basicin käytössä olevan muistialueen ylärajan osoite. Käyttöjärjestelmä ja Basic-tulkki käyttävät näiden muistipaikkojen si-



sältöä hyväkseen sijoittaessaan Basic-ohjelmaa ja sen tuottamia muuttujia muistiin. Konetta voi kuitenkin huijata luulemaan, että Basicin käytössä on vähemmän muistia pienentämällä tämän muistipaikan arvoa. Muuttamalla tätä osoitetta Basic-ohjelman alussa, voimme varata tilaa konekieliohjelmalle. Basicista käsin se käy seuraavalla käskyparilla:

**10 POKE 106,PEEK(106)-4\*X**

**20 GRAPHICS 0**

jossa X tarkoittaa tällä tavalla ryöstettyjen kilotavujen määrää. GRAPHICS-käsky asettaa uudelleen kuvamuistin. Nollan sijaan voi toki asettaa minkä tahansa grafiikkatilan 0-63. Osoitteessa 106 sijaitsee yläpään osoitteen enemmän merkitsevä tavu eli jokainen yksikkö siinä vastaa 256 muistipaikkaa.

Jos menemme vielä pidemmälle tilan varaamisessa, voimme aukaista 24 kilotavua RAM-muistia kytkemällä Basic-tulkin ja käyttöjärjestelmän sisältävän ROM-muistin pois päältä. Sen voisimme tehdä Basicistakin käsin muuttamalla muistipaikan \$D301 (54017) sisältöä sopivasti, mutta siinä ei olisi mitään mieltä, koska seuraavaa Basic-käskyä ei enää suoritettaisi. Se on sen vuoksi tehtävä konekielellä, esimerkiksi seuraavasti:

**\$8000 LDA \$D301 (lataa akkuun muistipaikan \$D301 sisältö)**

**\$8003 AND #\$FF (maskaa bitti 1 ykköseksi)**

**\$8005 STA \$D301 (vie tulos muistipaikkaan \$D301)**

**\$8008 ...**

Basic-tulkin poistamisella saadaan lisätilaa, mutta sen jälkeen ei voi enää käyttää hyväkseen Basicissa sijaitsevia valmisrutiineja.

Sama koskee käyttöjärjestelmän poiskytkemistä, joka tehdään edellisen esimerkin mukaisesti, mutta muuttamalla AND-käskyn argumentiksi \$FC. Samanaikaisesti ne poistetaan argumentilla \$FE. Perustilassa muistipaikan \$D301 arvo on \$FD.

ROM-muistin poiston jälkeen konekieliohjelmamme on aivan omillaan. Mitään koneen valmisrutiineista ei ole käytettävissä. Ennen käyttöjärjestelmän poiskytkemistä on vielä varmistettava keskeytysten palveluiden jatkuminen, muuten tulee ongelmia.

Konekielirutiinit ovat harvoin niin pitkiä, ettei niille riittäisi 256 tavun täysin vapaa alue lisättynä tarpeellisella määrällä Basic-ohjelmilta lainattua muistia.

## Haetaan näppäimiltä

Konekieliohjelmienkin on haettava ohjausta käyttäjältään. Peleissä se useimmiten hoituu peliohjaimen välityksellä, mutta muissa tilanteissa luonnollisin yhteydenpitoväline on näppäimistö.

Jos ohjelmaa kiinnostaa ainoastaan tällä hetkellä painettava näppäin, voidaan tieto alhaalla olevasta näppäimestä hakea osoitteesta \$2FC (764). Jos siellä on lu-

ku 255, ei mitään näppäintä paineta, muuten se sisältää painetun näppäimen koodin.

Atarissa näppäimistöllä on yhden merkin pituinen puskuri. Se tarkoittaa, että alhaalla olevan näppäimen koodi voidaan lukea muistipaikasta 764 ja viimeksi painetun ja hyväksytyn näppäimen koodi muistiinpaikasta 754, vaikkei lukuhetkellä mitään näppäintä enää painettaisikaan. Näppäimistön koodit eivät ole minkäänlaisessa suhteessa ATASCII-koodiin. Näppäimistön kooditaulukko on ohessa.

SHIFT-näppäimen painaminen lisää muistipaikan arvoa 64:llä ja CONTROL-näppäin 128:lla

## ATARIN NÄPPÄINKOODIT

| näppäin | koodi | näppäin     | koodi |
|---------|-------|-------------|-------|
| A       | 63    | 3           | 26    |
| B       | 21    | 4           | 24    |
| C       | 18    | 5           | 29    |
| D       | 58    | 6           | 27    |
| E       | 42    | 7           | 51    |
| F       | 56    | 8           | 53    |
| G       | 61    | 9           | 48    |
| H       | 57    | 0           | 50    |
| I       | 13    | +           | 6     |
| J       | 1     | —           | 14    |
| K       | 5     | <           | 54    |
| L       | 0     | >           | 55    |
| M       | 37    | *           | 7     |
| N       | 35    | ;           | 2     |
| O       | 8     | =           | 15    |
| P       | 10    | ,           | 32    |
| Q       | 47    | .           | 34    |
| R       | 40    | /           | 38    |
| S       | 62    | ESC         | 28    |
| T       | 45    | TAB         | 44    |
| U       | 11    | DEL         | 52    |
| V       | 16    | RETURN      | 12    |
| W       | 46    | CAPS        | 60    |
| X       | 22    | KÄÄNTEIS    | 39    |
| Y       | 43    | VÄLILYÖNTI  | 33    |
| Z       | 23    | EI PAINETTU | 255   |
| 1       | 31    | SHIFT       | +64   |
| 2       | 30    | CONTROL     | +128  |

## Luetaan peliohjainta

Atarissa on kaksi peliohjainporttia. Ne sijaitsevat osoitteessa \$D300 (54016), josta käyttöjärjestelmä päivittää ne joka 50. sekunti muistipaikkoihin 632, 633. Näiden muistipaikkojen neljä alinta bittiä kuvaavat peliohjaimen kytkimiä seuraavasti:

|         |            |
|---------|------------|
| bitti 3 | oikealle   |
| bitti 2 | vasemmalle |
| bitti 1 | taakse     |
| bitti 0 | eteen      |

Normaalitilassa bitit ovat ykkösiä. Jos sauvaa käännetään vasemmalle muuttuu bitti 2 nolaksi, jos sauvaa käännetään oikealle eteen, muuttuvat bitit 0 ja 3 noliksi, jos sauvaa käännetään alas vasemmalle, muuttuvat bitit 1 ja 2 nolliksi jne.

Tulitusnäppäimet luetaan GTIA -piirin muistipaikoista 53264 (D010) ja 53265 (D010), joista käyttöjärjestelmä päivittää ne osoitteisiin 644 ja 645. Seuraavaa ohjelmanpätkää voit käyttää omissa ohjelmissasi testaamaan peliohjaimen 0 asentoa ja haarautumaan sen mukaan:

|        |            |                                                                                                                                                      |
|--------|------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$8000 | LDA \$0278 | (hae peliohjaimen 0 bitit akkuun)                                                                                                                    |
| \$8003 | LSR        | (siirrä C-lippuun ETEEN-bitti)                                                                                                                       |
| \$8004 | BCS \$8009 | (jos C=1, jatketaan ilman toimenpiteitä)                                                                                                             |
| \$8006 | JSR \$8100 | (muuten käydään ensin ETEEN-rutiinissa)                                                                                                              |
| \$8009 | LSR        | (siirrä C-lippuun TAAKSE-bitti)                                                                                                                      |
| \$800A | BCS \$800F | (jos C=1, jatketaan ilman toimenpiteitä)                                                                                                             |
| \$800C | JSR \$8200 | (muuten käydään ensin TAAKSE-rutiinissa)                                                                                                             |
| \$800F | LSR        | (siirrä C-lippuun VASEMMALLE-bitti)                                                                                                                  |
| \$8010 | BCS \$8015 | (jos C=1, jatketaan ilman toimenpiteitä)                                                                                                             |
| \$8012 | JSR \$8300 | (muuten käydään ensin VASEMMALLE-rutiinissa)                                                                                                         |
| \$8015 | LSR        | (siirrä C-lippuun OIKEALLE-bitti)                                                                                                                    |
| \$8016 | BCS \$801B | (jos C=1, jatketaan ilman toimenpiteitä)                                                                                                             |
| \$8018 | JSR \$8400 | (muuten käydään ensin OIKEALLE-rutiinissa)                                                                                                           |
| \$801B | LDA \$0284 | (hae TULITUS-näppäimen rekisteri akkuun)                                                                                                             |
| \$801E | BNE \$8023 | (jos rekisterin arvo $\diamond$ 0, jatketaan ilman toimenpiteitä)                                                                                    |
| \$8020 | JSR \$8400 | (muuten käydään ensin TULITUS-rutiinissa)                                                                                                            |
| \$8023 | ...        | (ohjelma jatkuu...)                                                                                                                                  |
| \$8100 | ...        | (aliohjelma, joka suorittaa siirron eteenpäin. Tässä ja seuraavissa aliohjelmissa on akku tallennettava, jotta muutkin ohjausbitit voitaisiin lukea) |
| \$81FF | RTS        | (paluu pääohjelmaan)                                                                                                                                 |
| \$8200 | ...        | (aliohjelma, joka suorittaa siirron taaksepäin)                                                                                                      |
| \$82FF | RTS        | (paluu pääohjelmaan)                                                                                                                                 |
| \$8300 | ...        | (aliohjelma, joka suorittaa siirron vasemmalle)                                                                                                      |

|               |            |                                               |
|---------------|------------|-----------------------------------------------|
| <b>\$83FF</b> | <b>RTS</b> | (paluu pääohjelmaan)                          |
| <b>\$8400</b> | <b>...</b> | (aliohjelma, joka suorittaa siirron oikealle) |
| <b>\$84FF</b> | <b>RTS</b> | (paluu pääohjelmaan)                          |
| <b>\$8500</b> | <b>...</b> | (aliohjelma, joka suorittaa tulituksen)       |
| <b>\$85FF</b> | <b>RTS</b> | (paluu pääohjelmaan)                          |

## Piirrämme kuvaruudulle

Atarissa kuvaruututulostukseen vaikuttaa kolme tekijää: kuvamuistin sisältö, värirekisterit ja videoprosessorin ohjaus. Värirekistereiden käyttö on kattavasti esitetty laitteen mukana tulevassa kirjassa Basic 1, joten keskitymme tämän kirjan puitteissa ainoastaan kahteen jälkimmäiseen.

Basicissa normaali tapa saada kuvaruudulla jotain aikaan on käyttää PRINT-lauseita. PRINT-lauseen lainausmerkkien sisällä voi olla ruudulle tulostettavia merkkejä ja kursorin eli kohdistimen siirtoja, jotka määräävät, mihin kohtaan kuvaruutua tulostus tapahtuu.

Basicilla toimittaessa voimme ohjata kuvaruutua myös suoraan POKE-käskyillä. Kaikki se, mitä kuvaruudulla näkyy, on sijoitettuna kuvamuistiin, jonka ensimmäisen tavun osoite on muistipaikoissa 88 ja 89.

Kuvamuistin ensimmäiset 40 paikkaa sisältävät ensimmäisen vaakarivin merkit, seuraavat 40 toisen vaakarivin jne. Rivejä on 24, joista koostuu yhteensä 960 muistipaikkaa. POKE-käskyllä voimme sijoittaa minkä merkin tahansa mihin kohtaan kuvaruutua tahansa.

Atarissa voi kuvamuistin sijoittaa mihin tahansa muistissa. Lisäksi videoprosessorin ohjelman muuttaminen mahdollistaa jopa eri grafiikkatilojen sekoittamisen, joten kuvamuistin alkuosoite ei ole itsestään selvä. Kuvamuistin osoite on löydettävissä komennolla PEEK(88)\*256+PEEK(89). Koneen käynnistämisen ja RESET-näppäimen painamisen jälkeen (grafiikkatilassa 0) se alkaa muistiosoitteessa 40000 ja videoprosessorin (Antic) ohjelma osoitteessa 39968.

Kokeile heti käynnistämisen jälkeen vaikkapa seuraavaa:

**POKE 40500,56**  
**POKE 39968,48**  
**POKE 39972,65**  
**POKE 39973,157**

Ensimmäinen komento kirjoittaa kuvaruudun keskelle X-kirjaimen, toinen täryttää sitä hiukan ylöspäin muuttamalla Anticin ohjelmaa. Myös kolmas ja neljäs POKE muuttavat Anticin ohjelmaa. Ne siirtävät X-kirjainta muuttamalla videoprosessorin näkemystä kuvamuistin alkuosoitteesta.

Tarkemmin selitettynä ensimmäinen komento asetti kuvamuistiin X-kirjaimen sisäisen koodin. Tällä tavoin voi kuvaruudulle kirjoittaa pitempiäkin tekstejä. Samalla tavoin voimme kirjoittaa konekielellä. POKE-käsky vain vaihdetaan ko-

nekielen käskypariin LDA ja STA. LDA:lla ladataan ensin akkuun halutun merkin kuvaruutukoodi ja STA:lla se viedään oikeaan paikkaan.

Sinun on itse laskettava halutun kuvaruutupaikan osoite ja vietävä koodi sinne STA:lla. Koneen valmisrutiineihin kuuluva CIO (Central I/O) tarjoaa toisen mahdollisuuden kuvaruudulle tulostamiseen. Tämä tekniikka selvitetään myöhemmin tekstissä.

Jos haluat kirjoittaa ruudulle pitempiä viestejä tai muita merkkisarjoja, ei ole mielekästä ladata jokaista erikseen akkuun ja siitä siirtää ruudulle. Kätevintä on sijoittaa se ja muut vastaavat merkkijonot kuvaruutukoodien jonoina konekielisen ohjelman ulkopuolelle, alkuun tai loppuun, ja rakentaa aliohjelma, joka ilmoitetusta osoitteesta lähtien tulostaa ruudulle merkkijonon halutusta osoitteesta alkaen.

Koodijonon ensimmäiseksi tavuksi voi laittaa kyseisen jonon pituuden, jolloin aliohjelman ei tarvitse etukäteen tietää edes tulostettavan jonon pituutta. Saman asian voi toteuttaa myös sijoittamalla jokaisen koodijonon viimeiseksi tavuksi \$FF (255) tai \$DB (219, Return), jonka tulostusaliohjelma sitten tunnistaa ja lopettaa siirtämisen.

Tässä esimerkki mahdollisesta aliohjelmasta, joka siirtää annetusta osoitteesta lähtien koodijonoa kuvaruudulle. Ennen aliohjelman kutsumista on koodijonon alkuosoite sijoitettava osoitteisiin \$CB ja \$CC, vähemmän merkitsevä tavu ensin mainittuna. Kohdeosoite kuvaruudulla on sijoitettava vastaavasti paikkoihin \$CD ja \$CE. Nämä osoitteet ovat käyttäjälle vapaata tilaa nollasivulla ja soveltuvat erinomaisesti, koska niiden yhteydessä voimme käyttää epäsuoria indeksoituja osoitustapoja. Siirrettävän merkkijonon ensimmäinen tavu sisältää siirrettävien tavujen lukumäärän eli merkkijonon pituuden.

|        |              |                                                         |
|--------|--------------|---------------------------------------------------------|
| \$0600 | LDY #\$00    | ;sijoita Y-rekisteriin nolla                            |
| \$0602 | LDA (\$CB),Y | ;hae ensimmäinen merkki eli merkkien lukumäärä          |
|        |              | ;akkuun                                                 |
| \$0604 | TAY          | ;siirrä se Y-rekisteriin                                |
| \$0605 | LDA (\$CB),Y | ;hae merkki akkuun, viimeinen merkki ensin. \$CB:ssä    |
|        |              | ;ja \$CC:ssä oli ennalta asetettu koodijonon alkuosoite |
| \$0607 | STA (\$CD),Y | ;vie se kuvaruudulle. \$CD:ssä ja \$CE:ssä oli ennalta  |
|        |              | ;asetettu kuvaruudun osoite                             |
| \$0609 | DEY          | ;seuraava merkki lopusta alkuunpäin lukien              |
| \$060A | BNE \$0605   | ;oliko viimeinen eli oliko Y nolla?                     |
| \$060C | RTS          | ;jos oli, palataan rutiinista                           |

## Atarin kuvaruutukoodit

ATASCII-koodit vastaavat ASCII-koodeja kirjaimien, numeroiden ja välimerkkien osalta. ATASCII-koodistossa eivät koodit 0-31 kuitenkaan ole kontrolli-

merkkejä vaan grafiikkamerkkejä. Lisäksi merkitsevin bitti on pariteetin tarkastuksen sijaan otettu merkitsemään käänteistä esitysmuotoa.

Atarissa videoprosessori tulkitsee nollan välilyönniksi eli tyhjäksi. Tämä siksi, että kuvaruudun tyhjentäminen on helpointa nollaamalla kyseinen muistialue. ASCII- ja ATASCII-koodistossa välilyönnin arvo on 32.

| ATASCII-koodi | Kuvaruutu-koodi |
|---------------|-----------------|
| 0- 31         | ATASCII+64      |
| 32- 95        | ATASCII-32      |
| 96-127        | ATASCII         |
| 128-159       | ATASCII+64      |
| 160-223       | ATASCII-32      |
| 224-255       | ATASCII         |

POKEamalla tai viemällä STA-käskyllä kuvaruudulle koodiluvun saat ruudulle sitä vastaavan merkin. Luvut 128-255 tuottavat samat merkit kuin luvut 0-127, mutta käänteisinä, eli taustan ja merkin värit vaihdettuna keskenään. Merkin voit siis muuttaa käänteiseksi lisäämällä koodilukuun 128.

## Grafiikan hallinta ja Antic

Atarin grafiikka kuuluu parhaisiin, mitä kotimikroissa on nähty. Syntyessään (v.1977) poikkesi sen videopiiri, Antic huomattavasti muiden mikrojen vastaavista. Se tarjosi mahdollisuuden itsensä ohjelmointiin. Vasta viime vuosina on mikroissa (Apple Macintosh, Amiga) nähty sen ominaisuudet ylittäviä toteutuksia.

Antic on itse asiassa mikroprosessori 6502:n tapaan. Sillä on oma käskykanta, jonka ohjaamana se pitää yllä näyttöä. Anticin ohjelmaa kutsutaan display listiksi (DL), joka voidaan vapaasti suomentaa kuvamuistin tulkintaohjeeksi.

Anticin käskykanta on huomattavasti yksinkertaisempi kuin 6502:n ja samalla se on erittäin tiukasti suunnattu sen omaan tehtävään, kuvaruudun hallintaan. Ohjelma koostuu pääasiassa listasta grafiikkatiloja, joiden mukaan kuvamuistia tulkitaan.

Kussakin grafiikkatilassa kuvaruudun rivin täyttämiseen vaaditaan määrätty määrä kuvamuistin tavuja. Ennen kunkin kuvaruudun rivin lukemista Antic lukee ohjelmastaan tulevan rivin grafiikkatilan ja lähettää sen ja tarvittavan määrän tavuja GTIA-piirille, joka muokkaa näistä tiedoista television kuvan.

Grafiikkatilan tai -tilojen ja kuvamuistin osoitteen tai osoitteiden (kuva voidaan todellakin lukea vaikka jokainen rivi omasta kuvamuististaan) lisäksi DL voi sisältää näytön vaaka- ja pystysuunnan vierityksen sallimisia tai estoja, hypykäskyjä, käskyjä ohittaa joitain kuvaruudun juovia tulostamatta niille mitään, keskeytyspyyntöjä ja käskyn odottaa keskeytystä.

Perustilassa DL sijaitsee välittömästi kuvamuistin alapuolella. Sen osoite on löydettävissä muistipaikoista \$230 ja \$231 (560 ja 561), joista käyttöjärjestelmä

päivittää sen Anticin muistialuelle osoitteisiin \$D402 ja \$D403 (54274 ja 54275).

Yleisin DL:n muunnelmä on parin merkkigrafiikkarivin lisääminen bittikar-toitetun grafiikkaruudun ylä- tai alareunaan. Tällä tempulla on useimpien pe-lien alareunaan saatu rivit pisteitä ja peliaikaa varten.

Seuraava POKE syötettynä heti koneen käynnistämisen jälkeen, muuttaa DL:n kahdeksatta tavua. Sen vaikutuksesta kuvaruudun kolmas rivi tulkitaan grafiik-katilan 1 mukaan. Kokeile muitakin Anticin grafiikkatiloja 2-15.

### POKE 39975,6

Atarissa on yhdeksän valmiiksi ohjelmoitua kuvamuistin tulkintaohjelmaa. Esi-merkiksi otamme grafiikkatilan kaksi, koska sen ohjelma on lyhin. Ohjelma al-kaa muistipaikassa 40536 (GRAPHICS 2 -komennon jälkeen). Anticin käskytavut on selvitetty tarkemmin seuraavilla sivuilla.

#### käskytavu selitys

|      |                                                                           |
|------|---------------------------------------------------------------------------|
| \$70 | Kukin rivi määrää 8 tyhjää juovaa tulostumaan                             |
| \$70 | kuvaruudun yläreunaan (yhteensä 24). \$00 määräisi yhden juovan,          |
| \$70 | \$10 kaksi juovaa jne.                                                    |
| \$47 | $\$40 + \$7 \Rightarrow$ uusi kuvamuistin osoite = rivi graf. 2:n mukaan. |
| \$70 | Kuvamuistin ensimmäisen rivin alkuosoite                                  |
| \$9E | on \$9E70 (40560)                                                         |
| \$7  | Kukin rivi vastaa yhtä grafiikkatilan 2                                   |
| \$7  | mukaan tulkittua riviä. Yhdelle riville mahtuu 20                         |
| \$7  | merkkiä, joten osoitteesta \$9E70 alkaen luetaan                          |
| \$7  | $10 \times 20 = 200$ tavua.                                               |
| \$7  |                                                                           |
| \$7  |                                                                           |
| \$7  |                                                                           |
| \$7  | Kaksi riviä jää teksti-ikkunan alle, joten niitä                          |
| \$7  | ei näy tulkintaohjeissa.                                                  |
| \$42 | $\$40 + \$2 \Rightarrow$ uusi kuvamuistin osoite + rivi graf. 0:n mukaan  |
| \$60 | Teksti-ikkunan ensimmäisen rivin alkuosoite on \$9F60                     |
| \$9F | (40800) eli välittömästi jatkona graf 2:n kuvamuistille.                  |
| \$2  | Yhteensä neljä riviä                                                      |
| \$2  | teksti-ikkunaa                                                            |
| \$2  | grafiikkatilassa 0.                                                       |
| \$41 | $\$40 + \$1 \Rightarrow$ ei grafiikkaa, vain hyppy + odota kuvaruudun     |
| \$58 | pystypalautusta (VBI). Anticin ohjelmalaskuriin ladataan                  |
| \$9E | \$9E58 (40536) ja ohjelman suoritus alkaa alusta.                         |

*Kuvaruudun tulkintaohjeet (DL) grafiikkatilassa 2.*

## Anticin käskykanta

DL:n toiminnan ymmärtämiseksi on tunnettava pari perusasiaa TV:n kuvanpiirrosta. TV:n kuvassa on 625 vaakajuovaa, joista vuorotellen puolet päivitetään. Päivitys tapahtuu 50 kertaa sekunnissa, joten jokainen juova päivitetään kerran joka 1/25 sekunti. Suuri osa juovista jää TV:n ruudun ulkopuolelle, jonka vuoksi Atarin grafiikan tarkkuudeksi on asetettu vain 192 vaakajuovaa. Videoprosessori lähettää TV:lle kummankin kuvaruudun piirron aikana saman kuvan, joten kuvaruudulla on aina kaksi riviä, joilla on sama sisältö.

Eri grafiikkatilat kuvaavat muistin sisällön näytölle erilailla. Display list määrittää tavan, jolla kuvamuistin osien sisältö tulkitaan erivärisiksi kuvaruudun pisteiksi.

DL:n käskytavujen merkitys tulkitaan kussakin tavussa päällä (1) olevien bittien mukaan. Bitit 0-3 määräävät seuraavan kuvaruudun rivin grafiikkatilan. Näistä kolmas bitti määrää onko kyseessä piste (1) vai merkkigrafiikka (0). Oheisen taulukon avulla voi muodostaa Anticin käskytavun alemman puolittavun.

### merkkigrafiikat

| Anticin grafiikka-tila | Basicin grafiikka-tila | Merkin koko<br>leveys*vaakajuovat<br>(värit) | tavuja rivillä |
|------------------------|------------------------|----------------------------------------------|----------------|
| \$2                    | 0                      | 8* 8 (2)                                     | 40             |
| \$3                    | -                      | 8*10 (2)                                     | 40             |
| \$4                    | 12                     | 8* 8 (4)                                     | 40             |
| \$5                    | 13                     | 8*16 (4)                                     | 40             |
| \$6                    | 1                      | 16* 8 (5)                                    | 20             |
| \$7                    | 2                      | 16*16 (5)                                    | 20             |

### pistegrafiikat

| Anticin grafiikka-tila | Basicin grafiikka-tila | Merkin koko<br>leveys*vaakajuovat<br>(värit) | tavuja rivillä |
|------------------------|------------------------|----------------------------------------------|----------------|
| \$8                    | 3                      | 8*8 (4)                                      | 10             |
| \$A                    | 5                      | 4*4 (4)                                      | 20             |
| \$D                    | 7                      | 2*2 (4)                                      | 40             |
| \$9                    | 4                      | 4*4 (2)                                      | 10             |
| \$B                    | 6                      | 2*2 (2)                                      | 20             |
| \$F                    | 8                      | 1*1 (2)                                      | 40             |
| \$C                    | 14                     | 2*1 (2)                                      | 20             |
| \$E                    | 15                     | 2*1 (4)                                      | 40             |

### muut

\$0 tulostaa vaakajuovia ilman grafiikkaa

\$1 Anticin hyppykäsky (jump)



Anticin käskyn bitit 0-3 ja niitä vastaavat Basicin grafiikkatilat resoluution ja tyyppin mukaisessa järjestyksessä. Anticin ohjelmaan sijoitettuna kukin ensimmäisen sarakkeen luvuista aiheuttaa kuvamuistissa seuraavana vuorossa olevien tavujen tulkitsemisen lukua vastaavan grafiikkatilan mukaan.

Grafiikkatilat 9-11 puuttuvat taulukosta, koska ne toteutetaan GTIA-piirin (Georg's Tv Interface Adapter) avulla. Anticin grafiikalla \$3 ei ole vastinetta Basicissa.

Anticin käskyn bitit 4 ja 5 sallivat tai kieltävät kuvan vierityksen pysty- ja vaakasuunnassa. Pystysuuntainen viritys sallitaan lisäämällä grafiikkatilaan \$10 (16) ja vaakasuuntainen lisäämällä \$20 (32).

Mikäli bitti 6 on päällä (+\$40) tulkitsee Antic kaksi seuraavaa tavua DL:ssä uudeksi kuvamuistin osoitteeksi. Sovelluksen bittien 4-6 käytöstä löytää esimerkiksi kilpa-ajopeleissä Electra Glide ja Pole Position. Näissä vuorimaiseman ja pilvien liike on toteutettu vierittämällä omaa muistilohkoa vaakasuunnassa ympäri (bitti viisi on päällä).

Erikoisasemassa on käsky \$41 (65), jolla merkitään kuvaruudun loppua. Tämän käskyn kohdatessaan Antic lataa ohjelmanaskuriinsa kahden seuraavan tavun sisältämän osoitteen ja jää odottamaan kuvaruudun pystypalautuksen synkronointipulssia ja sitä seuraavaa keskeytystä ennen kuin jatkaa ohjelmansa suorittamista. Tämä käsky sisältyy jokaiseen DL:iin. Miltei poikkeuksetta hyppy kohdistuu DL:n ensimmäiseen tavuun.

Bitti 7 (+\$80) asettaa keskeytyspyynnön (DLI=Display List Interrupt) kuvamuistin tulkintaohjelmaan. Keskeytyksen aikana voidaan suorittaa kuvaruudun leveydestä (muistipaikka \$22F, 559) riippuen 18-50 kellojakson pituinen konekieli-ohjelma. Tämä ohjelma voidaan suorittaa vaikka ennen jokaisen rivin tulostamista eli 0-192 kertaa kuvaruudun piirron aikana; parhaimmillaan siis 9600 kertaa sekunnissa (grafiikkatila 8).

Seuraavan esimerkin ensimmäinen käsky sallii Anticin ohjelman sisältämän keskeytyspyynnön ja toinen asettaa sen grafiikan 0 DL:n kahdeksanteen tavuun (128+2=130). Keskeytysohjelma suoritetaan tällöin ennen kuvaruudun toisen rivin piirtoa.

**POKE 54286,192**

**POKE 39975,130**

Keskeytyksen aikana suoritettavan konekieli-ohjelman osoite on ladattava muistipaikkoihin \$200 ja \$201 (512 ja 513). Käynnistämisen jälkeen nämä muistipaikat osoittavat RTI-käskyä, joten yllämainitut POKEt yksinään syötettynä eivät näytä tekevän mitään.

Bitin 7 yleisimmät sovellukset ovat taustan värin vaihto (esimerkiksi grafiikkas-  
sa 8 voi saada näkyviin 192 väriä vaihtamalla väriä jokaisen rivin jälkeen), musi-

kin soitto, spritejen liikuttelu, DL:n muuttaminen ja eräiden grafiikan vektoreiden manipulointi.

Mikäli DL:n käskytavussa ei ole määrätty grafiikkatilaa tulkitsee Antic sen käskyksi tulostaa tyhjiä juovia. \$00 merkitsee yhtä tyhjää juovaa, \$10 kahta, \$20 kolmea, ... ja \$70 (112) kahdeksaa juovaa. Juovat tulostuvat kuvaruudulle taustan värillä. Käsky niiden tuottamiseen voi sijaita missä tahansa kohtaa DL:ä. Grafiikkattoman juovan piirto ei muuta Anticin kuvamuistin osoitinta.

## Omat grafiikan tulkintaohjelmat

Oman tulkintaohjelman luominen edellyttää kohtuullista yhteenlaskun hallintaa. Jotta tulos olisi asiallinen on vaakajuovien määrän oltava 192. Kunkin grafiikkatilan yhden rivin vaakajuovien lukumäärä löytyy taulukossa, jossa selvitetään Anticin käskyn bittien 0-3 vaikutukset.

Sekoitetun grafiikan kuvamuistin sijoittamista ei käyttöjärjestelmä hallitse, joten se jää käyttäjän tehtäväksi. Mikäli kuvamuistille varataan liian vähän tilaa, löytää Antic runsaasti mielenkiintoisia eri grafiikkatiloissa tulkittavia merkkejä Basic-ROMin puolelta (eli suttua).

Tarkkaavainen lukija varmasti huomasi, että Anticilla ei ole grafiikkatilaa \$1. Tämä käsky kuitenkin kuuluu sen käskykantaan. Kyseessä on hyppykäsky, jolla Antic jatkaa DL:n suorittamista kahden seuraavan tavun osoittamasta muistipaikasta. Käskyä joudutaan toisinaan käyttämään rakennettaessa hyvin pitkää DL:ä.

Anticin osoiteväylä on nimittäin vain kymmenen bitin levyinen, joten se ei kykene laskemaan 1K:n rajan yli. Siksi sen käskykantaan on liitetty hyppykäsky (\$1), jolla se voi hypätä lohkon rajan yli.

## Syöttö ja tulostus — IOCB

Atarissa on syöttö- ja tulostustoiminnot rakennettu erilaisten "laitteiden" ympärille. Laitteiksi Atari ymmärtää muunmuassa näppäimistön (K:), kuvaruudun (S:), kasettiaseman (C:), levyaseman (D:), printterin (P:) ja ennenkaikkea editorin eli näppäimistön ja kuvaruudun yhdistelmän (E:). Basicissa näitä käsitellään avaamalla niihin kanava (esimerkiksi OPEN #1,4,0,C:). Kanavia on kaikkiaan kahdeksan, #0-#7. Kanavan #0 käyttöjärjestelmä varaa aina editorille (E:), Basicin GRAPHICS-käsky avaa kanavan #6 näytölle (S:) ja LPRINT #7:n kirjoittimelle (P:).

Konekielellä työskennellessä käytetään kanavista nimitystä komentolohko, IOCB (Input Output Command Block). Niitä on kanavien tapaan myös kahdeksan kappaletta ja niiden numerointi vastaa Basicin kanavien numerointia. Komentolohkoille on varattu muistialue \$340-\$3BF (832-959). Kun tämä jaetaan kahdeksalla saadaan kunkin pituudeksi 16 tavua.

Komentolohkot ovat eräänlaisia taulukoita, joihin määrättyihin paikkoihin sopivia arvoja sijoittamalla voi helposti välittää käyttöjärjestelmälle tiedot mo-

nimutkaisistakin syöttö- ja tulostustoiminnoista. Koko rakennelman idea piilee käyttöjärjestelmän laiteriippumattomassa CIO-rutiinissa (Central Input Output). Komentolohkon sisältämän datan avulla pystyy CIO toimimaan posteljoonina muistin ja halutun laitteen välillä.

CIO-rutiinia kutsuttaessa (JSR \$E456) välitetään komentolohkon numero X-rekisterin ylemmässä puolittavussa. CIO tutkii kyseisen komentolohkon ja suorittaa sen edellyttämät toimenpiteet. Komentolohkon datoista se selvittää minkä laitteen ja muistialueen välillä tieto siirretään, millaista tietoa siirretään, kuinka paljon ja luonnollisesti tiedonsiirron suunnan. Nämä saatuaan suorittaa CIO itse loput ja raportoi päätteeksi tiedonsiirron onnistumisesta.

Kuten Basicissa on konekielessäkin kanava avattava ennen käyttöä ja suljettava se käytön jälkeen. Konekielellä työskenneltäessä kannattaa komentolohko lisäksi sulkea ennen avaamista. Näin välttyy virhetilanteelta, joka syntyy kun yritetään avata jo avoinna oleva komentolohko.

Yleensä komentolohkot 1-5 ovat vapaana. Niitä voit siis käyttää. Seuraavassa on esimerkkinä esitetty komentolohkon #1 rakenne. Muut seitsemän lohkoa ovat rakenteeltaan samanlaisia. Komentolohkon 16 muistipaikasta pääosasta huolehtii käyttöjärjestelmä.

|                    |            |                                                                                                                                                                                                                           |
|--------------------|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$350<br>(848)     | LAITETAULU | Käyttöjärjestelmän asettama laitetaulukon numero, johon komentolohko on avoinna.                                                                                                                                          |
| \$351<br>(849)     | NUMERO     | Käyttöjärjestelmän asettama laitteen numero, kuten D2: tai P2:.                                                                                                                                                           |
| \$352<br>(850)     | KOMENTO    | Käyttäjän asetettavissa oleva halutun toiminnan numero.<br>\$03=avaus (open)<br>\$04=merkkijonon luku<br>\$07=binaaritiedon luku<br>\$08=merkkijonon kirjoitus<br>\$0B=binaaritiedon kirjoitus<br>\$0C=sulkeminen (close) |
| \$353<br>(851)     | TILA       | I/O-toiminnon jälkeen on status eli tila luettavissa tästä. Mikäli tila=1 onnistui I/O-toiminto.                                                                                                                          |
| \$354-5<br>(852-3) | LAITE      | Osoite, joka viittaa komentolohkoa avattaessa laitteen nimeen (C:, D:TIEDOSTO, E:, K:). Esimerkiksi editoria käytettäessä osoite viitattava tavuihin \$45 ja \$3A (E ja :).                                               |

|                                     |                                                                                                                                                                                                                         |
|-------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>PUSKURI</b>                      | Tietoa siirrettäessä on \$354-\$355:n viitattava tiedon-<br>siirrossa käytettävään puskurin alkuosoitteeseen, eli<br>muistialueeseen, jolta tieto siirretään laitteelle tai jon-<br>ne tieto laitteelta siirretään.     |
| <b>\$356-7 SIIRTO</b><br>(854-5)    | Käyttöjärjestelmän asettama kirjoitusrutiinin osoite.<br>Vain Basic-tulkki käyttää tätä.                                                                                                                                |
| <b>\$358-9 PITUUS</b><br>(856-7)    | Tiedonsiirrossa käytettävän puskurin pituus. Asetetaan<br>siirrettävän merkkijonon suurimman mahdollisen<br>pituuden mukaan.                                                                                            |
| <b>\$35A SUUNTA</b><br>(858)        | Tiedonsiirron suunta:<br>\$04=Luku (read)<br>\$06=Tiedostoluettelon luku (vain levyasema)<br>\$08=Kirjoitus (write)<br>\$09=kirjoitus, liitä aikaisemman tiedoston perään<br>(vain levyasema)<br>\$0C=Luku ja kirjoitus |
| <b>\$35B-F MUUINFO</b><br>(859-863) | Lisämääreitä asettavia tavuja. Basic-tulkki käyttää<br>näitä muun muassa NOTE- ja POINT-komentojen<br>yhteydessä.                                                                                                       |

Atarissa kaikki I/O-toiminnot peliohjainten tutkimista lukuunottamatta ta-  
pahtuvat komentolohkojen ja CION välityksellä. I/O-toiminnot suoritetaan täyt-  
tämällä sopivasti haluttu IOCB, lataamalla sen numero X-rekisteriin ja hyppää-  
mällä CIO-rutiiniin. Mikäli I/O-toiminto epäonnistui jää tilarekisterin  
negatiivinen-lippu päälle. Oheiset kolme esimerkkiä selventävät CION ja komen-  
tolohkojen käyttöä.

```

$0600 LDX #$20 ;komentolohko #2
$0602 LDA #$0C ;lohkon sulkemiskomento
$0604 STA $0342,X ; tallennetaan muistipaikkaan $362.
$0607 JSR $E456 ;I/O-toimenpiteen suoritus

```

Komentokanavan (kanava #2) sulkeminen. Muista, että Mikroassembler varaa  
osan vapaista kanavista (#1 ja #2) käyttöönsä.

```

$060A LDX #$20 ;komentolohko #2
$060C LDA #$03 ;$03 on lohkon avauskomento
$060E STA $0342,X ; tallennetaan muistipaikkaan $342+$20=$362

```

```

$0611 LDA #$45 ;laitteeksi valitaan editori, $45="E"
$0613 STA $CB ; tallennus vapaaseen nollasivun paikkaan 203
$0615 LDA #$3A ; laitetunnukseen kuuluva ":" ($3A=":")
$0617 STA $CC ; tallennetaan osoitteeseen 204
$0619 LDA $CB ;osoitteisiin $364 ja $365 tallennetaan
$061B STA $0344,X ; käytettävän laitteen nimen osoite $00CB
$061E LDA $00 ; nimen osoitteen merkitsevämpi tavu
$0620 STA $0345,X ; tallennetaan muistipaikkaan $365
$0623 LDA #$08 ;$08 sallii I/O-yhteydelle vain kirjoituksen
$0625 STA $034A,X; suunta tallennetaan muistipaikkaan $36A
$0628 JSR $E456 ;I/O-toimenpiteen suoritus

```

Komentokanavan #2 avaaminen editorille (E:). Vaihtamalla muistipaikan \$0612 arvoksi \$50 (= "P") aloittaa CIO keskustelun printterin kanssa. Yleensä laitteiden nimet tallennetaan datana ohjelman loppuun.

```

$062B LDX #$20 ;komentolohko #2
$062D LDA #$08 ;$08 on merkkijonon kirjoituskomento
$062F STA $0342,X ; tallennetaan muistipaikkaan $362
$0632 LDA $4B ;merkkijonon osoite (50251)
$0634 STA $0344,X ; tallennetaan muistipaikkaan $364
$0637 LDA $C4 ; osoitteen merkitsevämpi tavu
$0639 STA $0345,X ; tallennetaan muistipaikkaan $365
$063C LDA $FF ;merkkien maksimi määrä
$063E STA $0348,X ; tallennetaan muistipaikkaan $368
$0641 LDA $00 ; merkkien määrän merkitsevämpi tavu
$0643 STA $0349,X ; tallennetaan muistipaikkaan $369
$0646 JSR $E456 ;I/O-toimenpiteen suoritus
$0649 RTS ;paluu Mikroassembleriin

```

ASCII-merkkijonon tulostus. Tulostuksen lopettaa EOL-merkki eli RETURN (\$9B, 155). Merkkien määrän (\$348-\$349) on oltava vähintään yhtä suurempi kuin tulostettavien merkkien määrä. Ohjelma kirjoittaa editorille käyttöjärjestelmässä olevan tekstin BOOT ERROR. Virheilmoituksen ja sitä seuraavan laitemäärittelyn E: (jota käyttämällä olisi riveistä \$0611-\$0617 päässyt eroon) voit myös lukea Basic-komennolla

**FOR I=50251 TO 50263:CHR\$(PEEK(I));NEXT I**

## Kirjallisuutta

MAPPING THE ATARI, *COMPUTE! Publications, Inc.*

DE RE ATARI, *Atari Inc., USA*

TECHNICAL USER,S NOTES, *Atari Inc., USA*

COMPUTE'S THIRD BOOK OF ATARI, *COMPUTE! Publications, Inc.*

## Atarin RAM-parametreja

Tietokoneen käyttöjärjestelmä ja Basic-tulkki pitävät RAM-muistin alapäässä tallessa useita koneen käytön kannalta oleellisia muuttujia, parametreja. Muutamat niistä ovat käyttökelpoisia, kun haluat koneen tekevän jotain omilla ehdoillasi.

Hyvä esimerkki tästä on Basic-ohjelmille varatun tilan loppuosoite (\$6A, 106), jota muuttamalla saat varattua vapaata tilaa konekielisille ohjelmille, sprite-grafiikalle ja omille merkkivalikoimille. Oheisessa muistikartassa on listattuna näistä muistipaikoista hyödyllisimmät ja sellaiset, joiden käyttö ei edellytä monimutkaisten käyttöjärjestelmätoimintojen hallitsemista.

Täydellisemmän luettelon näistä, samaten kuin käyttöjärjestelmän ja Basic-tulkkin tarjoamista valmisrutiineista, saat koneesi rakennetta ja toimintaa käsittelevästä hakuteoksesta, joita on lueteltu tämän luvun lopussa.

| osoite<br>heksana | osoite<br>10-järj. | kuvaus                                                                                                                                                                                                        |
|-------------------|--------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$0000            | 0                  | käyttöjärjestelmän käytössä ainoastaan käynnistytksen aikana                                                                                                                                                  |
| \$0001            | 1                  | Varattu itsetestiä varten. Mikäli tähän muistipaikkaan kirjoittaa arvon 0 siirtyy komento itsetestille RESETin painamisen jälkeen.                                                                            |
| \$0004-\$0005     | 4-5                | Levykkeen kolmen ensimmäisen sektorin eli latausohjelman tallennuksen alkuosoite (\$700, 1792).                                                                                                               |
| \$0009            | 9                  | Jos muistipaikan sisältö on yksi on ohjelman lataus levyltä onnistunut. Jos muistipaikan arvo on 255 lukkiutuu kone SYSTEM RESETiä painettaessa.                                                              |
| \$0012-\$0014     | 18-20              | kolmen tavun kellolaskuri, joka kasvaa yhdellä noin 1/50 sekunnin välein                                                                                                                                      |
| \$0055-\$0056     | 85-86              | kursorin sarake. Grafiikkatilasta riippuen sen maksimi-arvo vaihtelee 19-319                                                                                                                                  |
| \$0058-\$0059     | 88-89              | kuvamuistin alkuosoite                                                                                                                                                                                        |
| \$005D            | 93                 | kursorin alla oleva merkki                                                                                                                                                                                    |
| \$005E-\$005F     | 94-95              | kursorin senhetkinen muistiosoite                                                                                                                                                                             |
| \$0063            | 99                 | kursorin sijainti loogisella rivillä (0-119)                                                                                                                                                                  |
| \$006A            | 106                | käyttäjän käytössä olevan muistin yläraja 256 tavun lohkoina. Normaalisti (pl.600XL) muistipaikan arvo on 160 eli kuvamuisti päättyy muistipaikkaan 160*256-1=40959, ja Basic-tulkki alkaa osoitteesta 40960. |

|               |         |                                                                                                                                                                                                                                                                                                                                     |
|---------------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$0080-\$00CA | 128-202 | Basic-tulkin käyttämä RAM-alue                                                                                                                                                                                                                                                                                                      |
| \$0080-\$0081 | 128-129 | Basic-ohjelmatilan alkuosoite                                                                                                                                                                                                                                                                                                       |
| \$0082-\$0083 | 130-131 | Basicin muuttujataulukon alkuosoite                                                                                                                                                                                                                                                                                                 |
| \$0088-\$0089 | 136-137 | käyttäjän Basic-ohjelman alkuosoite. Ohjelma on tallennettu tokeneina.                                                                                                                                                                                                                                                              |
| \$0091-\$0092 | 144-145 | Basic-ohjelmalle varatun tilan loppuosoite                                                                                                                                                                                                                                                                                          |
| \$00CB-\$00D1 | 203-209 | VAPAA ALUE                                                                                                                                                                                                                                                                                                                          |
| \$00D4-\$00FF | 212-255 | liukulukulaskijan käyttämä alue. Tämä alue on käytettävissä mikäli ohjelma ei käytä liukulukuaritmetiikkaa eli kertolaskua, logaritmeja jne.                                                                                                                                                                                        |
| \$00D4-\$00D5 | 212-213 | Helppo tapa siirtää tietoa Basic-ohjelman ja konekielen välillä. Näiden muistipaikkojen sisältämä arvo (0-65335) sijoitetaan USR-lauseen dummy-muuttujan (A=USR(...)) arvoksi. Oman konekieliohjelman on kirjoitettava halutut arvot näihin muistipaikkoihin.                                                                       |
| \$0100-\$01FF | 256-511 | keskusyksikön pinomuisti                                                                                                                                                                                                                                                                                                            |
| \$0200-\$0229 | 512-553 | keskeytysvektorit                                                                                                                                                                                                                                                                                                                   |
| \$0200-\$0201 | 512-513 | alkuosoite ohjelmaan, joka suoritetaan ennen tietyn kuvaruudun vaakajuovan piirtämistä.<br>Keskeytyksen (DLI=Display List Interrupt) paikka merkitään videoprosessorin ohjelmaan kääntämällä kuvaruudun riviä vastaavan tavun seitsemäs bitti.<br>DLI:n sallimiseksi on muistipaikkaan \$D40E (54286) sijoitettava arvo 192 (\$C0). |
| \$0208-\$0209 | 520-521 | näppäimistön keskeytysvektori                                                                                                                                                                                                                                                                                                       |
| \$0222-\$0223 | 546-547 | osoite kuvaruudun piirron pystypalautuksen (VBI) aikana suoritettavaan ohjelmaan. Aikaa ohjelman suoritukseen on noin 3800 kellojaksoa. Suoritetaan jokaisen VBI:n aikana.                                                                                                                                                          |
| \$0224-\$0225 | 548-549 | osoite pystypalautuksen jälkeen suoritettavaan ohjelmaan. Aikaa on noin 20000 kellojaksoa.<br>Keskusyksikkö suorittaa, mikäli aikakriittisiä toimintoja (levy ja kasettitoiminnot ym. mp.\$42=0) ei ole meneillään.                                                                                                                 |
| \$22F         | 559     | Biteillä 0 ja 1 hallitaan kuva-alueen leveyttä.<br>1=>128 pikseliä, 2=>160 pikseliä(normaali) ja 3=192 pikseliä. Asettamalla bitit 2 ja 3 ykkösiksi sallitaan                                                                                                                                                                       |

|                      |                   |                                                                                                                                                                                                            |
|----------------------|-------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                      |                   | <b>P/M (sprite) -grafiikka. POKE 559,0 kytkee Antic-videoprosessorin pois päältä lisäten ohjelman suoritussnopeutta noin 30%, mutta pimentäen kuvaruudun. Osoitteen alkuarvo on 34.</b>                    |
|                      |                   | <b>Kokeile arvoja 0, 33 ja 35. Kahden jälkimmäisen käyttö vaatii kuvamuistin uudelleenjärjestelyä.</b>                                                                                                     |
| <b>\$0230-\$0231</b> | <b>560-561</b>    | <b>Videoprosessorin ohjelman alkuosoite.</b>                                                                                                                                                               |
| <b>\$02B6</b>        | <b>694</b>        | <b>0=normaalit merkit, \$80 (128)=käänteiset merkit. Näppäimistöltä tulevien merkkien ATASCIikodeille suoritetaan XOR tämän muistipaikan kanssa ennen kuvaruudulle tulostamista.</b>                       |
| <b>\$02F2</b>        | <b>754</b>        | <b>voimassa oleva näppäinpainallus näppäinkoodina, 255=ei paineta</b>                                                                                                                                      |
| <b>\$02FC</b>        | <b>764</b>        | <b>viimeisin näppäinpainallus näppäinkoodina</b>                                                                                                                                                           |
| <b>\$03C0-\$03E7</b> | <b>960-999</b>    | <b>printteripuskuri (40 tavua)</b>                                                                                                                                                                         |
| <b>\$03FD-\$047F</b> | <b>1021-1151</b>  | <b>kasettipuskuri (131 tavua)</b>                                                                                                                                                                          |
| <b>\$0480-\$04FF</b> | <b>1152-1279</b>  | <b>vapaa alue (128 tavua)</b>                                                                                                                                                                              |
| <b>\$0500-\$05FF</b> | <b>1280-1535</b>  | <b>liukulukulasimen työskentelytila. Vapaa alue mikäli liukulukuaritmetiikkaa ei käytetä (256 tavua).</b>                                                                                                  |
| <b>\$0600-\$06FF</b> | <b>1536-1791</b>  | <b>käyttäjän konekieliohjelmille varattu alue (256 tavua). Skandinaavinen ROM varaa muistipaikan 1791</b>                                                                                                  |
| <b>\$0700</b>        | <b>1792</b>       | <b>kasettiaseman käyttäjille Basic-alueen alku</b>                                                                                                                                                         |
| <b>\$0700-\$1D7B</b> | <b>1792-7547</b>  | <b>levykäyttöjärjestelmän muistiin sijoitettu osa</b>                                                                                                                                                      |
| <b>\$1540-\$3306</b> | <b>5440-13062</b> | <b>komennolla DOS muistiin latautuva osa levykäyttöjärjestelmästä.</b>                                                                                                                                     |
| <b>\$1D7C-\$9C1E</b> | <b>7548-39966</b> | <b>Basic-ohjelmalle vapaata tilaa grafiikkatilassa 0.</b>                                                                                                                                                  |
| <b>\$3307</b>        | <b>13062</b>      | <b>Tätä ylemmäs sijoitetut konekieliohjelmat voidaan helposti tallentaa levyille DOSin alla. Tätä alemmas sijoitetut (sivua 6 lukuunottamatta) jäävät DOSin alle, mikäli MEM.SAV-tiedostoa ei käytetä.</b> |



## Käyttöjärjestelmän varaamat alueet:

|                      |                    |                                                                                                                 |
|----------------------|--------------------|-----------------------------------------------------------------------------------------------------------------|
| <b>\$0700-\$9C1E</b> | <b>1792-39966</b>  | Basic-ohjelmille varattu tila kasettiaseman käyttäjillä.                                                        |
| <b>\$0700-\$1D7B</b> | <b>1792-7547</b>   | levykäyttöjärjestelmän muistiin sijoitettu osa                                                                  |
| <b>\$1540-\$3306</b> | <b>5440-13062</b>  | komennolla DOS muistiin latautuva levykäyttöjärjestelmän osa                                                    |
| <b>\$1D7C-\$9C1E</b> | <b>7548-39966</b>  | Basic-ohjelmille varattu tila käytettäessä levyasemaa.                                                          |
| <b>\$8000-\$9FFF</b> | <b>32768-40959</b> | 8kt RAM tai ROM moduuliportissa.                                                                                |
| <b>\$9C1F-\$9FFF</b> | <b>39967-40959</b> | kuvaruudun muistialue ja display list grafiikkatilassa 0.                                                       |
| <b>\$A000-\$BFFF</b> | <b>40960-49151</b> | Basic-tulkki (ROM) tai 8kt RAM                                                                                  |
| <b>\$C000-\$FFFF</b> | <b>49152-65535</b> | 4 kt käyttämätöntä (ei kuitenkaan käyttäjän käytettävissä), I/O-laitteet, merkkiROM ja KERNAL-ROM tai 16 kt RAM |

# Konekielen käskyt

Konekielen käskyt on tässä lueteltu aakkosjärjestyksessä. Taulukosta käy ilmi käytettävissä olevien osoitusmuotojen, käskykoodien, niiden pituuksien ja suoritusaikojen lisäksi myös niiden vaikutukset tilarekisterin lippuihin. Käskylyhenteet on ilmaistu siinä muodossa kuin tämän kirjan MikroASSEMBLER-ohjelma ne hyväksyy.

Käskyn osoiteosa eli argumentti on merkitty joko kirjaimilla \$NN tai \$NNNN. Kahta kirjainta NN käytetään, kun osoiteosaksi voi laittaa vain kaksinumeroisen heksaluvun, eli kun on kysymyksessä joko välitön tai nollasivun osoitusmuoto. Neljää kirjainta NNNN käytetään, kun osoiteosaksi voi laittaa nelinumeroisen heksaluvun eli täyden osoitteen. MikroASSEMBLER-ohjelmalle voit antaa argumentin joko heksalukuna tai kymmenjärjestelmän lukuna.

Käskykoodi on ilmaistu heksalukuna ilman edeltävää dollarimerkkiä. Käskyn pituus tarkoittaa koko käskyn pituutta argumentteineen 8 bitin tavuina. Aika jaksoina tarkoittaa käskyn suoritusaikaa prosessorin kellojaksoina.

Taulukon tietoihin on muutamia poikkeuksia. Nämä erikoistapaukset on merkitty taulukkoon suluissa olevalla numerolla. Merkkien selitykset ovat seuraavat:

- (1) suoritusaika on konejakson verran pitempi, jos sivuraja ylittyy
- (2) suoritusaika on konejakson verran pitempi, jos hyppy tapahtuu toiselle sivulle
- (3) Nollalippu eli Z-lippu ei toimi oikein, jos D-lippu on asetettu eli laskenta tapahtuu kymmenjärjestelmän mukaan. Akku on testattava erikseen nollatilan havaitsemiseksi

| <div> <div>ADC</div> <div>lisää akkuun muistinumeron kera<br/>Add With Carry (1) (3)</div> </div> |              |                  |                 |               |
|---------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                     | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| välitön                                                                                           | ADC # \$NN   | 69               | 2               | 2             |
| absoluuttinen                                                                                     | ADC \$NNNN   | 6D               | 3               | 4             |
| nollasivu                                                                                         | ADC \$NN     | 65               | 2               | 3             |
| nollasivu,X                                                                                       | ADC \$NN,X   | 75               | 2               | 4             |
| absoluuttinen,X                                                                                   | ADC \$NNNN,X | 7D               | 3               | 4             |
| absoluuttinen,Y                                                                                   | ADC \$NNNN,Y | 79               | 3               | 4             |
| epäsuora,X                                                                                        | ADC (\$NN,X) | 61               | 2               | 6             |
| epäsuora,Y                                                                                        | ADC (\$NN),Y | 71               | 2               | 5             |
| <div> <div>vaikutukset<br/>tilarekisteriin</div> <div>N V * * * Z C</div> </div>                  |              |                  |                 |               |

| <div> <div>AND</div> <div>JA-toiminto akun ja muistipaikan kesken<br/>And Accumulator With Memory (1)</div> </div> |              |                  |                 |               |
|--------------------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                                      | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| välitön                                                                                                            | AND# \$NN    | 29               | 2               | 2             |
| absoluuttinen                                                                                                      | AND \$NNNN   | 2D               | 3               | 3             |
| nollasivu                                                                                                          | AND \$NN     | 25               | 2               | 2             |
| nollasivu,X                                                                                                        | AND \$NN,X   | 35               | 2               | 2             |
| absoluuttinen,X                                                                                                    | AND \$NNNN,X | 3D               | 3               | 3             |
| absoluuttinen,Y                                                                                                    | AND \$NNNN,Y | 39               | 3               | 3             |
| epäsuora,X                                                                                                         | AND (\$NN,X) | 21               | 2               | 2             |
| epäsuora,Y                                                                                                         | AND (\$NN),Y | 31               | 2               | 2             |
| <div> <div>vaikutukset<br/>tilarekisteriin</div> <div>N * * * * Z *</div> </div>                                   |              |                  |                 |               |

| <b>ASL</b> Siirto vasemmalle yhden bitin verran<br>Arithmetic Shift Left                       |              |                  |                 |               |
|------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                  | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| absoluuttinen                                                                                  | ASL \$NNNN   | 0E               | 3               | 6             |
| nollasivu                                                                                      | ASL \$NN     | 06               | 2               | 5             |
| akku                                                                                           | ASL          | 0A               | 1               | 2             |
| nollasivu,X                                                                                    | ASL \$NN,X   | 16               | 2               | 6             |
| absoluuttinen,X                                                                                | ASL \$NNNN,X | 1E               | 3               | 7             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> N       *       *       *       *       Z       C |              |                  |                 |               |

| <b>BCC</b> Haarautuminen, jos C-lippu = 0<br>Branch on Carry Clear (2)                         |              |                  |                 |               |
|------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                  | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| suhteellinen                                                                                   | BCC \$NNNN   | 90               | 2               | 3             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> *       *       *       *       *       *       * |              |                  |                 |               |

| <b>BCS</b> Haarautu, jos C-lippu = 1 (muistinumero)<br>Branch on Carry Set (2) |              |                  |                 |               |
|--------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                  | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| suhteellinen                                                                   | BCS \$NNNN   | B0               | 2               | 3             |
| vaikutukset<br>tilarekisteriin           *   *   *   *   *   *   *             |              |                  |                 |               |

| <b>BEQ</b> Haarautu, jos tulos oli nolla (Z=1)<br>Branch if Equal (2) |              |                  |                 |               |
|-----------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                         | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| suhteellinen                                                          | BEQ \$NNNN   | F0               | 2               | 3             |
| vaikutukset<br>tilarekisteriin           *   *   *   *   *   *   *    |              |                  |                 |               |

| <b>BIT</b> Vertaa muistipaikan bittejä akun bitteihin<br>Test Bits in Memory with Accumulator |              |                  |                 |               |
|-----------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                 | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| absoluuttinen                                                                                 | BIT \$NNNN   | 2C               | 3               | 4             |
| nollasivu                                                                                     | BIT \$NN     | 24               | 2               | 3             |
| vaikutukset<br>tilarekisteriin           N   V   *   *   *   Z   *                            |              |                  |                 |               |

| <b>BMI</b> Haarautu, jos tulos negatiivinen (N=1)<br>Branch on Result Minus (2) |              |                  |                 |               |
|---------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                   | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| suhteellinen                                                                    | BMI \$NNNN   | 30               | 2               | 3             |
| vaikutukset<br>tilarekisteriin *   *   *   *   *   *   *                        |              |                  |                 |               |

| <b>BNE</b> Haarautu, jos tulos ei ole nolla (Z=0)<br>Branch on Result Not Zero (2) |              |                  |                 |               |
|------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                      | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| suhteellinen                                                                       | BNE \$NNNN   | D0               | 2               | 3             |
| vaikutukset<br>tilarekisteriin *   *   *   *   *   *   *                           |              |                  |                 |               |

| <b>BPL</b> Haarautu, jos tulos positiivinen (N=0)<br>Branch on Result Plus (2) |              |                  |                 |               |
|--------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                  | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| suhteellinen                                                                   | BPL \$NNNN   | 10               | 2               | 3             |
| vaikutukset<br>tilarekisteriin *   *   *   *   *   *   *                       |              |                  |                 |               |

| <b>BRK</b> Ohjelmallinen keskeytys<br>Break                                                        |              |                  |                 |               |
|----------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                      | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                          | BRK          | 00               | 1               | 7             |
| vaikutukset<br>tilarekisteriin                   *       *       B       *       I       *       * |              |                  |                 |               |

| <b>BVC</b> Haarautu, jos ylivuotolippu = 0 (V=0)<br>Branch on Overflow Clear (2)                   |              |                  |                 |               |
|----------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                      | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| suhteellinen                                                                                       | BVC \$NNNN   | 50               | 2               | 3             |
| vaikutukset<br>tilarekisteriin                   *       *       *       *       *       *       * |              |                  |                 |               |

| <b>BVS</b> Haarautu, jos ylivuotolippu = 1 (V=1)<br>Branch on Overflow Set (2)                     |              |                  |                 |               |
|----------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                      | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| suhteellinen                                                                                       | BVS \$NNNN   | 70               | 2               | 3             |
| vaikutukset<br>tilarekisteriin                   *       *       *       *       *       *       * |              |                  |                 |               |

| <b>CLC</b> Nollaa C-lippu (muistinumero)<br>Clear Carry Flag                                                |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                               | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                                   | CLC          | 18               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      *        *        *        *        *        *        C |              |                  |                 |               |

| <b>CLD</b> Nollaa D-lippu (binaari-BCD)<br>Clear Decimal Mode                                               |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                               | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                                   | CLD          | D8               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      *        *        *        D        *        *        * |              |                  |                 |               |

| <b>CLI</b> Nollaa I-lippu (keskeytyskieltolippu)<br>Clear Interrupt Disable                                 |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                               | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                                   | CLI          | 58               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      *        *        *        *        I        *        * |              |                  |                 |               |



| <b>CLV</b> Nollaa V-lippu (ylivuotolippu)<br>Clear Overflow Flag                       |              |                  |                 |               |
|----------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                          | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                              | CLV          | B8               | 1               | 2             |
| <b>vaikutukset</b><br>tilarekisteriin      *      V      *      *      *      *      * |              |                  |                 |               |

| <b>CMP</b> Vertaa akkua muistipaikan sisältöön<br>Compare Memory and Accumulator       |              |                  |                 |               |
|----------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                          | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| välitön                                                                                | CMP #\$\$NN  | C9               | 2               | 2             |
| absoluuttinen                                                                          | CMP \$NNNN   | CD               | 3               | 4             |
| nollasivu                                                                              | CMP \$NN     | C5               | 2               | 3             |
| nollasivu,X                                                                            | CMP \$NN,X   | D5               | 2               | 4             |
| absoluuttinen,X                                                                        | CMP \$NNNN,X | DD               | 3               | 4             |
| absoluuttinen,Y                                                                        | CMP \$NNNN,Y | D9               | 3               | 4             |
| epäsuora,X                                                                             | CMP (\$NN,X) | C1               | 2               | 6             |
| epäsuora,Y                                                                             | CMP (\$NN),Y | D1               | 2               | 5             |
| <b>vaikutukset</b><br>tilarekisteriin      N      *      *      *      *      Z      C |              |                  |                 |               |

| <b>CPX</b> Vertaa X-rekisterin sisältöä<br>Compare Memory and Index X                       |              |                  |                 |               |
|---------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                               | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| välitön<br>absoluuttinen<br>nollasivu                                                       | CPX #\$\$NN  | E0               | 2               | 2             |
|                                                                                             | CPX \$NNNN   | EC               | 3               | 4             |
|                                                                                             | CPX \$NN     | E4               | 2               | 3             |
| vaikutukset<br>tilarekisteriin <div>             N   *   *   *   *   Z   C           </div> |              |                  |                 |               |

| <b>CPY</b> Vertaa Y-rekisterin sisältöä<br>Compare Memory and Index Y                       |              |                  |                 |               |
|---------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                               | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| välitön<br>absoluuttinen<br>nollasivu                                                       | CPY #\$\$NN  | C0               | 2               | 2             |
|                                                                                             | CPY \$NNNN   | CC               | 3               | 4             |
|                                                                                             | CPY \$NN     | C4               | 2               | 3             |
| vaikutukset<br>tilarekisteriin <div>             N   *   *   *   *   Z   C           </div> |              |                  |                 |               |

| <b>DEC</b> Vähennä muistipaikan sisällöstä yksi<br>Decrement Memory by One                     |              |                  |                 |               |
|------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                  | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| absoluuttinen                                                                                  | DEC \$NNNN   | CE               | 3               | 6             |
| nollasivu                                                                                      | DEC \$NN     | C6               | 2               | 5             |
| nollasivu,X                                                                                    | DEC \$NN,X   | D6               | 2               | 6             |
| absoluuttinen,X                                                                                | DEC \$NNNN,X | DE               | 3               | 7             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> N       *       *       *       *       Z       * |              |                  |                 |               |

| <b>DEX</b> Vähennä X-rekisterin sisältöä yhdellä<br>Decrement Index X by One                   |              |                  |                 |               |
|------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                  | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                      | DEX          | CA               | 1               | 2             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> N       *       *       *       *       Z       * |              |                  |                 |               |

| <b>DEY</b> Vähennä Y-rekisterin sisältöä yhdellä<br>Decrement Index Y by One                   |              |                  |                 |               |
|------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                  | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                      | DEY          | 88               | 1               | 2             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> N       *       *       *       *       Z       * |              |                  |                 |               |

| <b>EOR</b> Ehdoton TAI akun ja muistipaikan välillä<br>Exclusive-OR Memory with Accumulator (1) |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                   | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| välitön                                                                                         | EOR #\$NN    | 49               | 2               | 2             |
| absoluuttinen                                                                                   | EOR \$NNNN   | 4D               | 3               | 4             |
| nollasivu                                                                                       | EOR \$NN     | 45               | 2               | 3             |
| nollasivu,X                                                                                     | EOR \$NN,X   | 55               | 2               | 4             |
| absoluuttinen,X                                                                                 | EOR \$NNNN,X | 5D               | 3               | 4             |
| absoluuttinen,Y                                                                                 | EOR \$NNNN,Y | 59               | 3               | 4             |
| epäsuora,X                                                                                      | EOR (\$NN,X) | 41               | 2               | 6             |
| epäsuora,Y                                                                                      | EOR (\$NN),Y | 51               | 2               | 5             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> N       *       *       *       *       Z       *  |              |                  |                 |               |

| INC                                                          |              | Lisää muistipaikan sisältöön yksi<br>Increment Memory by One |  |  |
|--------------------------------------------------------------|--------------|--------------------------------------------------------------|--|--|
| osoitusmuodot                                                | käskylyhenne | koodi<br>heksana                                             |  |  |
| absoluuttinen<br>nollasivu<br>nollasivu,X<br>absoluuttinen,X | INC \$NNNN   | æ*äjaks.                                                     |  |  |
|                                                              | INC \$NN     | 6                                                            |  |  |
|                                                              | INC \$NN,X   | 5                                                            |  |  |
|                                                              | INC \$NNNN,X | 6                                                            |  |  |
|                                                              |              | 7                                                            |  |  |
| vaikutukset<br>tilarekisteriin                               |              |                                                              |  |  |
| N * * * * Z *                                                |              |                                                              |  |  |

| <b>INX</b> Lisää X-rekisterin sisältöön yksi<br>Increment Index X by One                              |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                         | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                             | INX          | E8               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      N       *       *       *       *       Z       * |              |                  |                 |               |

| <b>INY</b> Lisää Y-rekisterin sisältöön yksi<br>Increment Index Y by One                              |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                         | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                             | INY          | C8               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      N       *       *       *       *       Z       * |              |                  |                 |               |

| <b>JMP</b> Hyppää uuteen osoitteeseen<br>Jump to New Location                                         |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                         | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| absoluuttinen<br>epäsuora                                                                             | JMP \$NNNN   | 4C               | 3               | 3             |
|                                                                                                       | JMP (\$NNNN) | 6C               | 3               | 5             |
| vaikutukset<br>tilarekisteriin                      *       *       *       *       *       *       * |              |                  |                 |               |

| <div>JSR</div> <div>Hyppää aliohjelmaan (paluusoite pinoon)<br/>Jump to Subroutine</div> |              |                  |                 |               |
|------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                            | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| absoluuttinen                                                                            | JSR \$NNNN   | 20               | 3               | 6             |
| <div>vaikutukset</div> <div>tilarekisteriin</div> <div>* * * * *</div>                   |              |                  |                 |               |

| <div>LDA</div> <div>Lataa akkuun muistipaikan sisältö<br/>Load Accumulator with Memory (1)</div> |              |                  |                 |               |
|--------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                    | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| välitön                                                                                          | LDA #\$NN    | A9               | 2               | 2             |
| absoluuttinen                                                                                    | LDA \$NNNN   | AD               | 3               | 4             |
| nollasivu                                                                                        | LDA \$NN     | A5               | 2               | 3             |
| nollasivu,X                                                                                      | LDA \$NN,X   | B5               | 2               | 4             |
| absoluuttinen,X                                                                                  | LDA \$NNNN,X | BD               | 3               | 4             |
| absoluuttinen,Y                                                                                  | LDA \$NNNN,Y | B9               | 3               | 4             |
| epäsuora,X                                                                                       | LDA (\$NN,X) | A1               | 2               | 6             |
| epäsuora,Y                                                                                       | LDA (\$NN),Y | B1               | 2               | 5             |
| <div>vaikutukset</div> <div>tilarekisteriin</div> <div>N * * * * Z *</div>                       |              |                  |                 |               |

| <b>LDX</b> <b>Lataa X-rekisteriin muistipaikan sisältö</b><br><b>Load Index X with Memory (1)</b> |               |                  |                 |               |
|---------------------------------------------------------------------------------------------------|---------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                     | käskylyhenne  | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| välitön                                                                                           | LDX # \$NN    | A2               | 2               | 2             |
| absoluuttinen                                                                                     | LDX \$NNNN    | AE               | 3               | 4             |
| nollasivu                                                                                         | LDX \$NN      | A6               | 2               | 3             |
| nollasivu, Y                                                                                      | LDX \$NN, Y   | B6               | 2               | 4             |
| absoluuttinen, Y                                                                                  | LDX \$NNNN, Y | BE               | 3               | 4             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> N      *      *      *      *      Z      *          |               |                  |                 |               |

| <b>LDY</b> <b>Lataa Y-rekisteriin muistipaikan sisältö</b><br><b>Load Index Y with Memory (1)</b> |               |                  |                 |               |
|---------------------------------------------------------------------------------------------------|---------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                     | käskylyhenne  | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| välitön                                                                                           | LDY # \$NN    | A0               | 2               | 2             |
| absoluuttinen                                                                                     | LDY \$NNNN    | AC               | 3               | 4             |
| nollasivu                                                                                         | LDY \$NN      | A4               | 2               | 3             |
| nollasivu, X                                                                                      | LDY \$NN, X   | B4               | 2               | 4             |
| absoluuttinen, X                                                                                  | LDY \$NNNN, X | BC               | 3               | 4             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> N      *      *      *      *      Z      *          |               |                  |                 |               |

| <b>LSR</b> Siirto oikealle yhden bitin verran<br>Logic Shift Right              |              |                  |                 |               |
|---------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                   | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| absoluuttinen                                                                   | LSR \$NNNN   | 4E               | 3               | 6             |
| nollasivu                                                                       | LSR \$NN     | 46               | 2               | 5             |
| akku                                                                            | LSR          | 4A               | 1               | 2             |
| nollasivu,X                                                                     | LSR \$NN,X   | 56               | 2               | 6             |
| absoluuttinen,X                                                                 | LSR \$NNNN,X | 5E               | 3               | 7             |
| vaikutukset<br>tilarekisteriin      N      *      *      *      *      Z      C |              |                  |                 |               |

| <b>NOP</b> Ei toimintaa<br>No Operation                                         |              |                  |                 |               |
|---------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                   | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                       | NOP          | EA               | 1               | 2             |
| vaikutukset<br>tilarekisteriin      *      *      *      *      *      *      * |              |                  |                 |               |



| <b>ORA</b> TAI-operaatio akun ja muistipaikan välillä<br>OR Memory with Accumulator            |              |                  |                 |               |
|------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                  | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| välitön                                                                                        | ORA # \$NN   | 09               | 2               | 2             |
| absoluuttinen                                                                                  | ORA \$NNNN   | 0D               | 3               | 4             |
| nollasivu                                                                                      | ORA \$NN     | 05               | 2               | 3             |
| nollasivu,X                                                                                    | ORA \$NN,X   | 15               | 2               | 4             |
| absoluuttinen,X                                                                                | ORA \$NNNN,X | 1D               | 3               | 4             |
| absoluuttinen,Y                                                                                | ORA \$NNNN,Y | 19               | 3               | 4             |
| epäsuora,X                                                                                     | ORA (\$NN,X) | 01               | 2               | 6             |
| epäsuora,Y                                                                                     | ORA (\$NN),Y | 11               | 2               | 5             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> N       *       *       *       *       Z       * |              |                  |                 |               |

| <b>PHA</b> Siirrä akun sisältö pinoon<br>Push Accumulator on Stack                             |              |                  |                 |               |
|------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                  | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                      | PHA          | 48               | 1               | 3             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> *       *       *       *       *       *       * |              |                  |                 |               |

| <b>PHP</b> Siirrä tilarekisterin sisältö pinoon<br>Push Processor Status on Stack                |              |                  |                 |               |
|--------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                    | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                        | PHP          | 08               | 1               | 3             |
| vaikutukset tilarekisteriin                    *       *       *       *       *       *       * |              |                  |                 |               |

| <b>PLA</b> Lataa akku pinosta<br>Pull Accumulator from Stack                                     |              |                  |                 |               |
|--------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                    | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                        | PLA          | 68               | 1               | 4             |
| vaikutukset tilarekisteriin                    N       *       *       *       *       Z       * |              |                  |                 |               |

| <b>PLP</b> Lataa tilarekisteri pinosta<br>Pull Processor Status from Stack                       |              |                  |                 |               |
|--------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                    | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                        | PLP          | 28               | 1               | 4             |
| vaikutukset tilarekisteriin                    N       V       B       D       I       Z       C |              |                  |                 |               |

| <b>ROL</b> Pyöritä vasemmalle yhden bitin verran<br>Rotate One Bit Left |              |                  |                 |               |
|-------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                           | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| absoluuttinen                                                           | ROL \$NNNN   | 2E               | 3               | 6             |
| nollasivu                                                               | ROL \$NN     | 26               | 2               | 5             |
| akku                                                                    | ROL          | 2A               | 1               | 2             |
| nollasivu,X                                                             | ROL \$NN,X   | 36               | 2               | 6             |
| absoluuttinen,X                                                         | ROL \$NNNN,X | 3E               | 3               | 7             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> N * * * * Z C              |              |                  |                 |               |

| <b>ROR</b> Pyöritä oikealle yhden bitin verran<br>Rotate One Bit Right |              |                  |                 |               |
|------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                          | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| absoluuttinen                                                          | ROR \$NNNN   | 6E               | 3               | 6             |
| nollasivu                                                              | ROR \$NN     | 66               | 2               | 5             |
| akku                                                                   | ROR          | 6A               | 1               | 2             |
| nollasivu,X                                                            | ROR \$NN,X   | 76               | 2               | 6             |
| absoluuttinen,X                                                        | ROR \$NNNN,X | 7E               | 3               | 7             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> N * * * * Z C             |              |                  |                 |               |

| <b>RTI</b> <b>Palaa keskeytysohjelmasta</b><br><b>Return from Interrupt</b>                     |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                   | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                       | RTI          | 40               | 1               | 6             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> <b>N      V      B      D      I      Z      C</b> |              |                  |                 |               |

| <b>RTS</b> <b>Palaa aliohjelmasta</b><br><b>Return from Subroutine</b>                          |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                   | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                       | RTS          | 60               | 1               | 6             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> <b>*      *      *      *      *      *      *</b> |              |                  |                 |               |

| <b>SBC</b> <b>Vähennä akusta lainanumeron kera</b><br><b>Subtract with Borrow (1)</b>           |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                   | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| välitön                                                                                         | SBC # \$NN   | E9               | 2               | 2             |
| absoluuttinen                                                                                   | SBC \$NNNN   | ED               | 3               | 4             |
| nollasivu                                                                                       | SBC \$NN     | E5               | 2               | 3             |
| nollasivu,X                                                                                     | SBC \$NN,X   | F5               | 2               | 4             |
| absoluuttinen,X                                                                                 | SBC \$NNNN,X | FD               | 3               | 4             |
| absoluuttinen,Y                                                                                 | SBC \$NNNN,Y | F9               | 3               | 4             |
| epäsuora,X                                                                                      | SBC (\$NN,X) | E1               | 2               | 6             |
| epäsuora,Y                                                                                      | SBC (\$NN),Y | F1               | 2               | 5             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> <b>N      V      *      *      *      Z      C</b> |              |                  |                 |               |

| <b>SEC</b> Aseta C-lippu (muistinumero)<br>Set Carry Flag                                             |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                         | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                             | SEC          | 38               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      *       *       *       *       *       *       C |              |                  |                 |               |

| <b>SED</b> Aseta D-lippu (binaari-BCD)<br>Set Decimal Mode                                            |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                         | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                             | SED          | F8               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      *       *       *       D       *       *       * |              |                  |                 |               |

| <b>SEI</b> Aseta I-lippu (keskeytyskielto)<br>Set Interrupt Flag                                      |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                         | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                             | SEI          | 78               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      *       *       *       *       I       *       * |              |                  |                 |               |

| <b>STA</b> Tallenna akku muistiin<br>Store Accumulator in Memory                                     |              |                  |                 |               |
|------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                        | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| absoluuttinen                                                                                        | STA \$NNNN   | 8D               | 3               | 4             |
| nollasivu                                                                                            | STA \$NN     | 85               | 2               | 3             |
| nollasivu,X                                                                                          | STA \$NN,X   | 95               | 2               | 4             |
| absoluuttinen,X                                                                                      | STA \$NNNN,X | 9D               | 3               | 5             |
| absoluuttinen,Y                                                                                      | STA \$NNNN,Y | 99               | 3               | 5             |
| epäsuora,X                                                                                           | STA (\$NN,X) | 81               | 2               | 6             |
| epäsuora,Y                                                                                           | STA (\$NN),Y | 91               | 2               | 6             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> *        *        *        *        *        *        * |              |                  |                 |               |

| <b>STX</b> Tallenna X-rekisteri muistiin<br>Store Index X in Memory                                  |              |                  |                 |               |
|------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                        | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| absoluuttinen                                                                                        | STX \$NNNN   | 8E               | 3               | 4             |
| nollasivu                                                                                            | STX \$NN     | 86               | 2               | 3             |
| nollasivu,Y                                                                                          | STX \$NN,Y   | 96               | 2               | 4             |
| <b>vaikutukset</b><br><b>tilarekisteriin</b> *        *        *        *        *        *        * |              |                  |                 |               |

| <b>STY</b> Tallenna Y-rekisteri muistiin<br>Store Index Y in Memory                                         |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                               | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| absoluuttinen                                                                                               | STY \$NNNN   | 8C               | 3               | 4             |
| nollasivu                                                                                                   | STY \$NN     | 84               | 2               | 3             |
| nollasivu,X                                                                                                 | STY \$NN,X   | 94               | 2               | 4             |
| vaikutukset<br>tilarekisteriin                      *        *        *        *        *        *        * |              |                  |                 |               |

| <b>TAX</b> Siirrä akun sisältö X-rekisteriin<br>Transfer Accumulator to Index X                             |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                               | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                                   | TAX          | AA               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      N        *        *        *        *        Z        * |              |                  |                 |               |

| <b>TAY</b> Siirrä akun sisältö Y-rekisteriin<br>Transfer Accumulator to Index Y                             |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                               | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                                   | TAY          | A8               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      N        *        *        *        *        Z        * |              |                  |                 |               |

| <b>TSX</b> Siirrä pino-osoitin X-rekisteriin<br>Transfer Stack Pointer to Index X                     |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                         | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                             | TSX          | BA               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      N       *       *       *       *       Z       * |              |                  |                 |               |

| <b>TXA</b> Siirrä X-rekisterin sisältö akkuun<br>Transfer Index X to Accumulator                      |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                         | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                             | TXA          | 8A               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      N       *       *       *       *       Z       * |              |                  |                 |               |

| <b>TXS</b> Siirrä X-rekisterin sisältö pino-osoittimeen<br>Transfer Index X to Stack Pointer          |              |                  |                 |               |
|-------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| osoitusmuodot                                                                                         | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                             | TXS          | 9A               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                      *       *       *       *       *       *       * |              |                  |                 |               |



|                                                                                                            |              |                  |                 |               |
|------------------------------------------------------------------------------------------------------------|--------------|------------------|-----------------|---------------|
| <div><div>TYA</div><div>Siirrä Y-rekisterin sisältö akkuun<br/>Transfer Index Y to Accumulator</div></div> |              |                  |                 |               |
| osoitusmuodot                                                                                              | käskylyhenne | koodi<br>heksana | pituus<br>tavua | aika<br>jaks. |
| sisältyvä                                                                                                  | TYA          | 98               | 1               | 2             |
| vaikutukset<br>tilarekisteriin                                                                             |              |                  |                 |               |
| N * * * * Z *                                                                                              |              |                  |                 |               |

# Lukujärjestelmä- muunnokset

Seuraavilla sivuilla on käytännöllinen taulukko lukujärjestelmämuunnoksista sekä konekielen käskylyhenteet koodien mukaisessa järjestyksessä.

Ensimmäinen pystyryvi on nollasta 255:een juokseva 10-järjestelmän luku. Seuraavassa sarakkeessa on vastaava heksaluku. Kolmannessa sarakkeessa on toisen sarakkeen heksaluku muunnettuna 10-järjestelmään olettaen, että heksaluku sijaitsee osoitteen enemmän merkitsevässä tavussa. Sen avulla voi nopeasti muuttaa nelinumeroisetkin heksaluvut 10-järjestelmään. Esimerkiksi \$E03A muunnetaan seuraavasti: katsotaan kolmannesta sarakkeesta \$E0:aa vastaava 10-järjestelmän luku, se on 57344. Siihen lisätään alemman tavun (\$3A) vastaava arvo ensimmäisestä sarakkeesta, 58, joten tulokseksi saadaan 57402.

Käskylyhenteiden argumentit on merkitty seuraavasti: \$NN tarkoittaa kaksi-numeroista heksalukua (esimerkiksi \$2A) ja \$NNNN tarkoittaa nelinumeroista heksalukua (esimerkiksi \$7FFF). Eri osoitusmuotojen merkintätavat löydät MikroASSEMBLER-ohjelmaa käsittelevästä liitteestä

| 10-järj. | heksaluku | MSB 10-j. | binaari-luku | käsky-lyhenne |
|----------|-----------|-----------|--------------|---------------|
| 0        | 00        | 0         | 00000000     | BRK           |
| 1        | 01        | 256       | 00000001     | ORA (\$NN,X)  |
| 2        | 02        | 512       | 00000010     | *             |
| 3        | 03        | 768       | 00000011     | *             |
| 4        | 04        | 1024      | 00000100     | *             |
| 5        | 05        | 1280      | 00000101     | ORA \$NN      |

| 10-järj. | heksa-luku | MSB 10-j. | binaari-luku | käsky-lyhenne |
|----------|------------|-----------|--------------|---------------|
| 6        | 06         | 1536      | 00000110     | ASL \$NN      |
| 7        | 07         | 1792      | 00000111     | *             |
| 8        | 08         | 2048      | 00001000     | PHP           |
| 9        | 09         | 2304      | 00001001     | ORA # \$NN    |
| 10       | 0A         | 2560      | 00001010     | ASL           |
| 11       | 0B         | 2816      | 00001011     | *             |
| 12       | 0C         | 3072      | 00001100     | *             |
| 13       | 0D         | 3328      | 00001101     | ORA \$NNNN    |
| 14       | 0E         | 3584      | 00001110     | ASL \$NNNN    |
| 15       | 0F         | 3840      | 00001111     | *             |
| 16       | 10         | 4096      | 00010000     | BPL \$NN      |
| 17       | 11         | 4352      | 00010001     | ORA (\$NN),Y  |
| 18       | 12         | 4608      | 00010010     | *             |
| 19       | 13         | 4864      | 00010011     | *             |
| 20       | 14         | 5120      | 00010100     | *             |
| 21       | 15         | 5376      | 00010101     | ORA \$NN,X    |
| 22       | 16         | 5632      | 00010110     | ASL \$NN,X    |
| 23       | 17         | 5888      | 00010111     | *             |
| 24       | 18         | 6144      | 00011000     | CLC           |
| 25       | 19         | 6400      | 00011001     | ORA \$NNNN,Y  |
| 26       | 1A         | 6656      | 00011010     | *             |
| 27       | 1B         | 6912      | 00011011     | *             |
| 28       | 1C         | 7168      | 00011100     | *             |
| 29       | 1D         | 7424      | 00011101     | ORA \$NNNN,X  |
| 30       | 1E         | 7680      | 00011110     | ASL \$NNNN,X  |
| 31       | 1F         | 7936      | 00011111     | *             |
| 32       | 20         | 8192      | 00100000     | JSR \$NNNN    |
| 33       | 21         | 8448      | 00100001     | AND (\$NN,X)  |
| 34       | 22         | 8704      | 00100010     | *             |
| 35       | 23         | 8960      | 00100011     | *             |
| 36       | 24         | 9216      | 00100100     | BIT \$NN      |
| 37       | 25         | 9472      | 00100101     | AND \$NN      |
| 38       | 26         | 9728      | 00100110     | ROL \$NN      |
| 39       | 27         | 9984      | 00100111     | *             |
| 40       | 28         | 10240     | 00101000     | PLP           |
| 41       | 29         | 10496     | 00101001     | AND # \$NN    |
| 42       | 2A         | 10752     | 00101010     | ROL           |
| 43       | 2B         | 11008     | 00101011     | *             |

| 10-järj. | heksa-luku | MSB 10-j. | binaari-luku | käsky-lyhenne |
|----------|------------|-----------|--------------|---------------|
| 44       | 2C         | 11264     | 00101100     | BIT \$NNNN    |
| 45       | 2D         | 11520     | 00101101     | AND \$NNNN    |
| 46       | 2E         | 11776     | 00101110     | ROL \$NNNN    |
| 47       | 2F         | 12032     | 00101111     | *             |
| 48       | 30         | 12288     | 00110000     | BMI \$NN      |
| 49       | 31         | 12544     | 00110001     | AND (\$NN),Y  |
| 50       | 32         | 12800     | 00110010     | *             |
| 51       | 33         | 13056     | 00110011     | *             |
| 52       | 34         | 13312     | 00110100     | *             |
| 53       | 35         | 13568     | 00110101     | AND \$NN,X    |
| 54       | 36         | 13824     | 00110110     | ROL \$NN,X    |
| 55       | 37         | 14080     | 00110111     | *             |
| 56       | 38         | 14336     | 00111000     | SEC           |
| 57       | 39         | 14592     | 00111001     | AND \$NNNN,Y  |
| 58       | 3A         | 14848     | 00111010     | *             |
| 59       | 3B         | 15104     | 00111011     | *             |
| 60       | 3C         | 15360     | 00111100     | *             |
| 61       | 3D         | 15616     | 00111101     | AND \$NNNN,X  |
| 62       | 3E         | 15872     | 00111110     | ROL \$NNNN,X  |
| 63       | 3F         | 16128     | 00111111     | *             |
| 64       | 40         | 16384     | 01000000     | RTI           |
| 65       | 41         | 16640     | 01000001     | EOR (\$NN,X)  |
| 66       | 42         | 16896     | 01000010     | *             |
| 67       | 43         | 17152     | 01000011     | *             |
| 68       | 44         | 17408     | 01000100     | *             |
| 69       | 45         | 17664     | 01000101     | EOR \$NN      |
| 70       | 46         | 17920     | 01000110     | LSR \$NN      |
| 71       | 47         | 18176     | 01000111     | *             |
| 72       | 48         | 18432     | 01001000     | PHA           |
| 73       | 49         | 18688     | 01001001     | EOR #\$NN     |
| 74       | 4A         | 18944     | 01001010     | LSR           |
| 75       | 4B         | 19200     | 01001011     | *             |
| 76       | 4C         | 19456     | 01001100     | JMP \$NNNN    |
| 77       | 4D         | 19712     | 01001101     | EOR \$NNNN    |
| 78       | 4E         | 19968     | 01001110     | LSR \$NNNN    |
| 79       | 4F         | 20224     | 01001111     | *             |
| 80       | 50         | 20480     | 01010000     | BVC \$NN      |
| 81       | 51         | 20736     | 01010001     | EOR (\$NN),Y  |

| 10-järj. | heksa-luku | MSB 10-j. | binaari-luku | käsky-lyhenne |
|----------|------------|-----------|--------------|---------------|
| 82       | 52         | 20992     | 01010010     | *             |
| 83       | 53         | 21248     | 01010011     | *             |
| 84       | 54         | 21504     | 01010100     | *             |
| 85       | 55         | 21760     | 01010101     | EOR \$NN,X    |
| 86       | 56         | 22016     | 01010110     | LSR \$NN,X    |
| 87       | 57         | 22272     | 01010111     | *             |
| 88       | 58         | 22528     | 01011000     | CLI           |
| 89       | 59         | 22784     | 01011001     | EOR \$NNNN,Y  |
| 90       | 5A         | 23040     | 01011010     | *             |
| 91       | 5B         | 23296     | 01011011     | *             |
| 92       | 5C         | 23552     | 01011100     | *             |
| 93       | 5D         | 23808     | 01011101     | EOR \$NNNN,X  |
| 94       | 5E         | 24064     | 01011110     | LSR \$NNNN,X  |
| 95       | 5F         | 24320     | 01011111     | *             |
| 96       | 60         | 24576     | 01100000     | RTS           |
| 97       | 61         | 24832     | 01100001     | ADC (\$NN,X)  |
| 98       | 62         | 25088     | 01100010     | *             |
| 99       | 63         | 25344     | 01100011     | *             |
| 100      | 64         | 25600     | 01100100     | *             |
| 101      | 65         | 25856     | 01100101     | ADC \$NN      |
| 102      | 66         | 26112     | 01100110     | ROR \$NN      |
| 103      | 67         | 26368     | 01100111     | *             |
| 104      | 68         | 26624     | 01101000     | PLA           |
| 105      | 69         | 26880     | 01101001     | ADC #\$NN     |
| 106      | 6A         | 27136     | 01101010     | ROR           |
| 107      | 6B         | 27392     | 01101011     | *             |
| 108      | 6C         | 27648     | 01101100     | JMP (\$NNNN)  |
| 109      | 6D         | 27904     | 01101101     | ADC \$NNNN    |
| 110      | 6E         | 28160     | 01101110     | ROR \$NNNN    |
| 111      | 6F         | 28416     | 01101111     | *             |
| 112      | 70         | 28672     | 01110000     | BVS \$NN      |
| 113      | 71         | 28928     | 01110001     | ADC (\$NN),Y  |
| 114      | 72         | 29184     | 01110010     | *             |
| 115      | 73         | 29440     | 01110011     | *             |
| 116      | 74         | 29696     | 01110100     | *             |
| 117      | 75         | 29952     | 01110101     | ADC \$NN,X    |
| 118      | 76         | 30208     | 01110110     | ROR \$NN,X    |
| 119      | 77         | 30464     | 01110111     | *             |

| 10-järj. | heksa-luku | MSB 10-j. | binaari-luku | käsky-lyhenne |
|----------|------------|-----------|--------------|---------------|
| 120      | 78         | 30720     | 01111000     | SEI           |
| 121      | 79         | 30976     | 01111001     | ADC \$NNNN,Y  |
| 122      | 7A         | 31232     | 01111010     | *             |
| 123      | 7B         | 31488     | 01111011     | *             |
| 124      | 7C         | 31744     | 01111100     | *             |
| 125      | 7D         | 32000     | 01111101     | ADC \$NNNN,X  |
| 126      | 7E         | 32256     | 01111110     | ROR \$NNNN,X  |
| 127      | 7F         | 32512     | 01111111     | *             |
| 128      | 80         | 32768     | 10000000     | *             |
| 129      | 81         | 33024     | 10000001     | STA (\$NN,X)  |
| 130      | 82         | 33280     | 10000010     | *             |
| 131      | 83         | 33536     | 10000011     | *             |
| 132      | 84         | 33792     | 10000100     | STY \$NN      |
| 133      | 85         | 34048     | 10000101     | STA \$NN      |
| 134      | 86         | 34304     | 10000110     | STX \$NN      |
| 135      | 87         | 34560     | 10000111     | *             |
| 136      | 88         | 34816     | 10001000     | DEY           |
| 137      | 89         | 35072     | 10001001     | *             |
| 138      | 8A         | 35328     | 10001010     | TXA           |
| 139      | 8B         | 35584     | 10001011     | *             |
| 140      | 8C         | 35840     | 10001100     | STY \$NNNN    |
| 141      | 8D         | 36096     | 10001101     | STA \$NNNN    |
| 142      | 8E         | 36352     | 10001110     | STX \$NNNN    |
| 143      | 8F         | 36608     | 10001111     | *             |
| 144      | 90         | 36864     | 10010000     | BCC \$NN      |
| 145      | 91         | 37120     | 10010001     | STA (\$NN),Y  |
| 146      | 92         | 37376     | 10010010     | *             |
| 147      | 93         | 37632     | 10010011     | *             |
| 148      | 94         | 37888     | 10010100     | STY \$NN,X    |
| 149      | 95         | 38144     | 10010101     | STA \$NN,X    |
| 150      | 96         | 38400     | 10010110     | STX \$NN,Y    |
| 151      | 97         | 38656     | 10010111     | *             |
| 152      | 98         | 38912     | 10011000     | TYA           |
| 153      | 99         | 39168     | 10011001     | STA \$NNNN,Y  |
| 154      | 9A         | 39424     | 10011010     | TXS           |
| 155      | 9B         | 39680     | 10011011     | *             |
| 156      | 9C         | 39936     | 10011100     | *             |
| 157      | 9D         | 40192     | 10011101     | STA \$NNNN,X  |

| 10-järj. | heksa-luku | MSB 10-j. | binaari-luku | käsky-lyhenne |
|----------|------------|-----------|--------------|---------------|
| 158      | 9E         | 40448     | 10011110     | *             |
| 159      | 9F         | 40704     | 10011111     | *             |
| 160      | A0         | 40960     | 10100000     | LDY #N        |
| 161      | A1         | 41216     | 10100001     | LDA (\$N,X)   |
| 162      | A2         | 41472     | 10100010     | LDX #N        |
| 163      | A3         | 41728     | 10100011     | *             |
| 164      | A4         | 41984     | 10100100     | LDY \$N       |
| 165      | A5         | 42240     | 10100101     | LDA \$N       |
| 166      | A6         | 42496     | 10100110     | LDX \$N       |
| 167      | A7         | 42752     | 10100111     | *             |
| 168      | A8         | 43008     | 10101000     | TAY           |
| 169      | A9         | 43264     | 10101001     | LDA #N        |
| 170      | AA         | 43520     | 10101010     | TAX           |
| 171      | AB         | 43776     | 10101011     | *             |
| 172      | AC         | 44032     | 10101100     | LDY \$NNN     |
| 173      | AD         | 44288     | 10101101     | LDA \$NNN     |
| 174      | AE         | 44544     | 10101110     | LDX \$NNN     |
| 175      | AF         | 44800     | 10101111     | *             |
| 176      | B0         | 45056     | 10110000     | BCS \$N       |
| 177      | B1         | 45312     | 10110001     | LDA (\$N),Y   |
| 178      | B2         | 45568     | 10110010     | *             |
| 179      | B3         | 45824     | 10110011     | *             |
| 180      | B4         | 46080     | 10110100     | LDY \$N,X     |
| 181      | B5         | 46336     | 10110101     | LDA \$N,X     |
| 182      | B6         | 46592     | 10110110     | LDX \$N,Y     |
| 183      | B7         | 46848     | 10110111     | *             |
| 184      | B8         | 47104     | 10111000     | CLV           |
| 185      | B9         | 47360     | 10111001     | LDA \$NNN,Y   |
| 186      | BA         | 47616     | 10111010     | TSX           |
| 187      | BB         | 47872     | 10111011     | *             |
| 188      | BC         | 48128     | 10111100     | LDY \$NNN,X   |
| 189      | BD         | 48384     | 10111101     | LDA \$NNN,X   |
| 190      | BE         | 48640     | 10111110     | LDX \$NNN,Y   |
| 191      | BF         | 48896     | 10111111     | *             |
| 192      | C0         | 49152     | 11000000     | CPY #N        |
| 193      | C1         | 49408     | 11000001     | CMP (\$N,X)   |
| 194      | C2         | 49664     | 11000010     | *             |
| 195      | C3         | 49920     | 11000011     | *             |

| 10-järj. | heksa-luku | MSB 10-j. | binaari-luku | käsky-lyhenne |
|----------|------------|-----------|--------------|---------------|
| 196      | C4         | 50176     | 11000100     | CPY \$NN      |
| 197      | C5         | 50432     | 11000101     | CMP \$NN      |
| 198      | C6         | 50688     | 11000110     | DEC \$NN      |
| 199      | C7         | 50944     | 11000111     | *             |
| 200      | C8         | 51200     | 11001000     | INY           |
| 201      | C9         | 51456     | 11001001     | CMP #\$NN     |
| 202      | CA         | 51712     | 11001010     | DEX           |
| 203      | CB         | 51968     | 11001011     | *             |
| 204      | CC         | 52224     | 11001100     | CPY \$NNNN    |
| 205      | CD         | 52480     | 11001101     | CMP \$NNNN    |
| 206      | CE         | 52736     | 11001110     | DEC \$NNNN    |
| 207      | CF         | 52992     | 11001111     | *             |
| 208      | D0         | 53248     | 11010000     | BNE \$NN      |
| 209      | D1         | 53504     | 11010001     | CMP (\$NN),Y  |
| 210      | D2         | 53760     | 11010010     | *             |
| 211      | D3         | 54016     | 11010011     | *             |
| 212      | D4         | 54272     | 11010100     | *             |
| 213      | D5         | 54528     | 11010101     | CMP \$NN,X    |
| 214      | D6         | 54784     | 11010110     | DEC \$NN,X    |
| 215      | D7         | 55040     | 11010111     | *             |
| 216      | D8         | 55296     | 11011000     | CLD           |
| 217      | D9         | 55552     | 11011001     | CMP \$NNNN,Y  |
| 218      | DA         | 55808     | 11011010     | *             |
| 219      | DB         | 56064     | 11011011     | *             |
| 220      | DC         | 56320     | 11011100     | *             |
| 221      | DD         | 56576     | 11011101     | CMP \$NNNN,X  |
| 222      | DE         | 56832     | 11011110     | DEC \$NNNN,X  |
| 223      | DF         | 57088     | 11011111     | *             |
| 224      | E0         | 57344     | 11100000     | CPX #\$NN     |
| 225      | E1         | 57600     | 11100001     | SBC (\$NN,X)  |
| 226      | E2         | 57856     | 11100010     | *             |
| 227      | E3         | 58112     | 11100011     | *             |
| 228      | E4         | 58368     | 11100100     | CPX \$NN      |
| 229      | E5         | 58624     | 11100101     | SBC \$NN      |
| 230      | E6         | 58880     | 11100110     | INC \$NN      |
| 231      | E7         | 59136     | 11100111     | *             |
| 232      | E8         | 59392     | 11101000     | INX           |
| 233      | E9         | 59648     | 11101001     | SBC #\$NN     |



| 10-järj. | heksa-luku | MSB 10-j. | binaari-luku | käsky-lyhenne |
|----------|------------|-----------|--------------|---------------|
| 234      | EA         | 59904     | 11101010     | NOP           |
| 235      | EB         | 60160     | 11101011     | *             |
| 236      | EC         | 60416     | 11101100     | CPX \$NNNN    |
| 237      | ED         | 60672     | 11101101     | SBC \$NNNN    |
| 238      | EE         | 60928     | 11101110     | INC \$NNNN    |
| 239      | EF         | 61184     | 11101111     | *             |
| 240      | F0         | 61440     | 11110000     | BEQ \$NN      |
| 241      | F1         | 61696     | 11110001     | SBC (\$NN),Y  |
| 242      | F2         | 61952     | 11110010     | *             |
| 243      | F3         | 62208     | 11110011     | *             |
| 244      | F4         | 62464     | 11110100     | *             |
| 245      | F5         | 62720     | 11110101     | SBC \$NN,X    |
| 246      | F6         | 62976     | 11110110     | INC \$NN,X    |
| 247      | F7         | 63232     | 11110111     | *             |
| 248      | F8         | 63488     | 11111000     | SED           |
| 249      | F9         | 63744     | 11111001     | SBC \$NNNN,Y  |
| 250      | FA         | 64000     | 11111010     | *             |
| 251      | FB         | 64256     | 11111011     | *             |
| 252      | FC         | 64512     | 11111100     | *             |
| 253      | FD         | 64768     | 11111101     | SBC \$NNNN,X  |
| 254      | FE         | 65024     | 11111110     | INC \$NNNN,X  |
| 255      | FF         | 65280     | 11111111     | *             |

# MikroASSEMBLER

MikroASSEMBLER on monipuolinen työkalu konekieliseen ohjelmointiin. Se sisältää kaikki tarvittavat perustoiminnot assemblerkielisen listauksen sisäänsyöttämiseen, ohjelman korjaamiseen, tutkimiseen, tallettamiseen ja lataamiseen.

MikroASSEMBLER (MA) sisältää itse assemblerin ohella disassemblerin, jolla voit omien ohjelmiesi lisäksi tutkia vaikkapa käyttöjärjestelmää tai Basic-tulkkia. Disassembler muuttaa muistissa olevan konekielisen ohjelman ruudulle (tai kirjoittimelle) takaisin samanlaisiksi kolmikirjaimisiksi käskyiksi, kuin millä ohjelmasi syötätkin.

MA:ssa on myös monitoriohjelma, jonka avulla voit tutkia yksittäisen muisti-paikan sisältöä ja muuttaa sitä. Samaten voit käynnistää konekielisen ohjelmasi, tallentaa sen levyille tai kasetille (mikä on ehdottomasti tehtävä ennen käynnistämistä) tai ladata levyltä/kasetilta aiemmin tekemäsi konekielisen ohjelman muistiin.

MA on kuitenkin perustyökalu, joka ei sisällä kaikkia ajateltavissa olevia toimintoja. Siitä puuttuu mm. "labelien" eli leimojen käyttömahdollisuus. Assembler-riveihin ja lukuihin ei siis voi viitata nimellä, vaan hyppykäskyissä on annettava kohdeosoite suoraan muistiosoitteena.

MA on tehty Basicilla. Se on kaikilta toiminnoiltaan silti riittävän nopea. Useat assembleriohjelmat on tehty konekielellä ja niiden nopeus tuo mukanaan muutamia etuja. MA riittää konekieliohjelmoijan tarpeisiin kuitenkin hyvin pitkälle. Kun olet konekieliohjelmoinnin ekspertti, voit harkita jonkin kaupallisen assembler/monitori -ohjelman hankkimista.

MA vie muistista tilaa vähän yli seitsemän kilotavua (VICin +3kt-versio laajimmillaan noin 5 kilotavua). Muistiin jää vielä senkin jälkeen tilaa pitkillekin

konekielisille ohjelmille.

Ennen kuin syötät ohjelman koneeseesi, tutustu liitteenä olevaan lukuun "Ohjelmien syöttö". Lukemalla sen tarkasti voit välttää syöttövirheitä, joita on jälkikäteen hankala jäljittää. C-64:n ja VIC-20:n osalta syöttöä helpottaa Tarkastajaniminen apuohjelma. Tarkastajan avulla voit ohjelmaa naputellessasi tarkkailla rivi riviltä mahdollisia syöttövirheitä.

## ASSEMBLER

Käynnistäessäsi ohjelman, tulee näkyviin aloitusruutu, jossa kerrotaan, kuinka eri toiminnot käynnistetään. Samalla ohjelma lataa taulukkoon DATA-riveiltä tiedot 6502:n käskykannasta. Kun näytön alalaitaan tulee teksti: "paina näppäintä", pääset ohjelman pääosaan, jossa voit syöttää koneeseen konekielisen ohjelmasi kirjassa esiintyvillä kolmikirjaimisilla lyhenteillä (mnemoniset käskyt).

Muut toiminnot (disassembler, monitori jne.) käynnistetään myös tästä pääohjelmasta käsin. VICin +3kt:n versiossa disassembleri ja monitori on erotettu omaksi ohjelmakseen. Disassembler käynnistetään antamalla käskyriville tavallisen konekielisen käskyn sijasta pistekoodi .DIS . Paluu muista toiminnoista tapahtuu samalle kohdalle käskyjen syöttövaiheeseen.

Kun olet aloitusruudun kehoituksesta painanut näppäintä, ohjelma kysyy alkuosoitetta. Kertaa, mitä edellisissä luvuissa on esitetty oman koneesi vapaista alueista, jonne konekieliset ohjelmat voi sijoittaa. Alkuosoitteen voit antaa joko heksalukuna, jolloin sen eteen on muistettava laittaa \$-merkki, tai kymmenjärjestelmän lukuna.

Kun olet antanut alkuosoitteen, kirjoittaa ohjelma seuraavan rivin alkuun antamasi osoitteen muutettuna heksaluvuksi ja siirtää cursorin hieman etäämmälle rivillä. Tähän kohtaan voit nyt kirjoittaa ohjelmasi ensimmäisen mnemonisen käskyn, esimerkiksi

**\$0340 LDA #\$2A (lataa akkuun luku \$2A (46))**

Ohjelma on tarkka välilyöntien suhteen. Ennen käskysanaa EI saa laittaa välilyöntiä (space), mutta käskysanan ja argumentin väliin on AINA pantava välilyönti. Poikkeuksena ovat käskyt, joilla ei ole argumenttia lainkaan, esimerkiksi CLC (nollaa C-lippu). Silloin ei käskysanan jälkeen saa laittaa välilyöntiä. Käskyjen jälkeen tulevia kommentteja MA ei tallenna. Niitä et siis voi ohjelmiisi lisätä.

Paina käskyrivin syötettyäsi RETURN-näppäintä. Ohjelma hakee antamaasi käskyä vastaavat koodit taulukoistaan ja sijoittaa ne rivinumeron ja mnemonisen käskyn väliin seuraavalla tavalla:

**\$0340 A9 2A LDA #\$2A**

Ohjelma sijoittaa käskykoodit myös suoraan muistiin, eli ohjelmasi rakentuu

muistissa todelliseksi ohjelmaksi sitä mukaa kuin käskyjä kirjoitat. Ohjelmasi on siis ajettavissa vaikka heti viimeisen käskyrivin syötön jälkeen (tallenna se kuitenkin ensin).

MA on tehty siten, että se pyrkii löytämään lähes kaikki syöttövirheet. Niistä se kommentoi kirjoittamalla rivin loppuun: VIRHE, VIRHE MNE tai VIRHE OSM. MNE tarkoittaa, että virhe on käskysanassa ja OSM, että virhe löytyy argumentista tai osoitusmuodosta. Aivan kaikkia virheitä ei MA kuitenkaan pysty löytämään, joten toimi huolellisesti ja pyri syöttämään käskysi oikein. Jos olet tuhrinut pahasti ja ohjelma 'heittää sinut ulos' virheilmoituksen kera, esimerkiksi:

**ILLEGAL QUANTITY ERROR IN 3040  
READY,**

niin kirjoita GOTO 345 ennen kuin teet mitään muuta. Ohjelma jatkuu suoraan käskyjen syötöstä.

MikroASSEMBLER esittää luvut ja koodit kuvaruudulla aina heksajärjestelmän mukaisesti. Kaikki tarvittavat luvut voit kuitenkin syöttää ohjelmalle myös kymmenjärjestelmän lukuina ("desimaalisina"), jos haluat. Silloin vain jätät luvun edestä \$-merkin pois. Seuraavat käskyt vastaavat siis toisiaan ja tuottavat samat koodit käskyriville:

**LDA #\$FF**

ja

**LDA #255**

Näin voit edetä käsky käskyltä ohjelman loppuun asti. Ohjelma kirjoittaa näytölle uuden muistiosoitteen, sinä kirjoitat haluamasi käskyrivin ja ohjelma taas laskee vastaavat koodit ja sijoittaa ne ruudulle sekä muistiin oikeille paikoilleen.

## Osoitusmuodot

Käskyjen eri osoitusmuodot ilmaistaan samalla tavalla kuin kirjan esimerkeissä on esitetty:

|                               |                     |                                                                  |
|-------------------------------|---------------------|------------------------------------------------------------------|
| <b>Välitön osoitus:</b>       | <b>LDA #\$4C</b>    | (lataa akkuun luku \$4C)                                         |
| <b>Absoluuttinen osoitus:</b> | <b>LDA \$2AF8</b>   | (lataa akku osoitteesta \$2AF8)                                  |
| <b>Nollasivun osoitus:</b>    | <b>LDA \$80</b>     | (lataa akku osoitteesta \$80)                                    |
| <b>Akkurekisteriosoitus:</b>  | <b>ASL</b>          | (ASL:llä on muitakin osoitusmuotoja)                             |
| <b>Sisältyvä osoitus:</b>     | <b>DEX</b>          | (vähennä X-rekisteriä yhdellä)                                   |
| <b>Epäsuora,X osoitus:</b>    | <b>LDA (\$80,X)</b> | (\$80:een lisätään X ennen käyttöä epäsuorana osoitteena)        |
| <b>Epäsuora,Y osoitus:</b>    | <b>LDA (\$80),Y</b> | (\$80:n ja \$81:n sisältöön lisätään Y ennen käyttöä osoitteena) |
| <b>Nollasivu,X osoitus:</b>   | <b>LDA \$45,X</b>   | (lataa akku osoitteesta \$45+X)                                  |

|                             |                     |                                                                     |
|-----------------------------|---------------------|---------------------------------------------------------------------|
| <b>Nollasivu,Y osoitus:</b> | <b>STX \$45,Y</b>   | (tallenna X-rekisteri osoitteeseen \$45+Y)                          |
| <b>Absoluuttinen,X:</b>     | <b>LDA \$2CF8,X</b> | (lataa akku osoitteesta \$2CF8+X)                                   |
| <b>Absoluuttinen,Y:</b>     | <b>LDA \$2CF8,Y</b> | (lataa akku osoitteesta \$2CF8+Y)                                   |
| <b>Epäsuora osoitus:</b>    | <b>JMP (\$2CF8)</b> | (hyppää osoitteeseen, joka löytyy muistipaikoista \$2CF8 ja \$2CF9) |

Kaikki osoitusmuodot eivät ole käytettävissä kaikkien käskyjen yhteydessä. Epäsuora osoitusmuoto toimii itse asiassa vain JMP:n kanssa ja Nollasivu,Y:tä voidaan käyttää ainoastaan LDX tai STX -käskyissä. Muitakin rajoituksia on ja ne selviävät parhaiten tapauskohtaisesti liitteenä olevasta käskytaulukosta.

Eräs osoitusmuoto jäi listasta vielä pois: suhteellinen osoitus. Se on toiminnaltaan mutkikkaampi ja sitä käsitellään erikseen seuraavassa kappaleessa.

## Hyppykäskyt

Ehdolliset hyppykäskyt (esimerkiksi BNE tai BEQ) ovat konekielessä kahden tavun pituisia. Ensimmäinen tavu kertoo, mistä käskystä on kyse ja toinen tavu kertoo, kuinka etäälle nykyisestä käskystä on hypättävä. Käsien koodattaessa tuon etäisyyden, "offsetin", laskeminen saattaa joskus olla hyvinkin hankalaa.

MA sen sijaan laskee sen puolestasi. Hyppykäskyn jälkeen annetaan vain se absoluuttinen osoite, johon halutaan hypätä, esimerkiksi

**\$0342      BNE \$0348      (hyppää osoitteeseen \$0348, mikäli tulos ei ollut nolla)**

MA laskee, kuinka etäällä tuo osoite sijaitsee ja sijoittaa oikean "offset"-arvon (\$04) muistiin ja kuvaruudulle BNE-käskyä vastaavan koodin D0 kanssa seuraa-vasti:

**\$0342 D0 04      BNE \$0348**

Jos hyppysi suuntautuu ohjelmassa taaksepäin, on tilanne selvä: katsot kohde-osoitteen halutun käskyrivin alusta, mikäli se vielä näkyy kuvaruudulla. Jos ei näy etkä ole kirjoittanut osoitetta muistiin, on sinun välillä käytävä disassemble-rin puolella listaamassa ohjelmaasi ruudulle ja etsittävä oikea osoite sieltä.

Jos hyppysi suuntautuu eteenpäin, on tilanne mutkikkaampi: ethän vielä tiedä, missä osoitteessa se käsky tulee sijaitsemaan, johon hyppäät. Nyt valittavissasi on kaksi vaihtoehtoa. Ensinnäkin voit yrittää tehdä mahdollisimman hyvän arvauksen ja kirjoitat sen myös paperille muistiin. Kun syöttövuorossa on kyseinen käsky, tarkistat paperilta, osuiko arvauksesi oikeaan (yleensä ei osu!).

Jos arvaus ei ollut oikea, on sinun käytävä korjaamassa hyppyosoite. Se tapahtuu pistekäskyn .ASM avulla.

Äskeisessä esimerkissä teimme BNE-hypyn osoitteeseen \$0348 ja rivin alussa oli osoite \$0342. Havaitsimme myöhemmin, että oikea hyppyosoite olisi ollut \$0349.

Kirjoita nyt:

|              |      |        |                                                                                                                                                                                                                           |
|--------------|------|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| \$0349       | .ASM | \$0342 | (olisimme juuri syöttäneet sen käskyn, johon piti hypätä, mutta päätimme käydä vielä muistaessamme ensin korjaamassa hyppyosoitteen. Kirjoittamalla .ASM \$0342 siirtyy "ohjelmalaskurimme" takaisin osoitteeseen \$0342) |
| \$0342 D0 05 | BNE  | \$0349 | (huomaa, että rivin alkuun ilmestyi nyt jälleen viallisen rivin osoite \$0342. Kirjoitimme sen perään BNE-käskyn uudelleen oikealla hyppyosoitteella)                                                                     |
| \$0344       | .ASM | \$0349 | (kun olemme korjanneet hyppyosoitteen, on meidän siirryttävä takaisin sinne, missä olimme käskyllä .ASM \$0349, muuten pyyhkisimme uusilla käskyillä yli kaikki välissä olevat jo ohjelmoidut käskyt)                     |
| \$0349       | ...  |        | (jatkamme ohjelmointia)                                                                                                                                                                                                   |

Toinen tapa hoitaa eteenpäin suuntautuvat hyppykäskyt on, ettemme edes yritä arvata kohdeosoitetta, vaan sijoitamme hyppyosoitteeksi seuraavan käskyn osoitteen. Edellisessä esimerkissä se olisi ollut BNE \$0344, koska hyppykäsky vie tilaa kaksi tavua, \$0342:n ja \$0343:n.

Voit jälkikäteen korjata keskitetysti hyppyosoitteet. Erotat tällaisen ratkaisemattoman hypyn siitä, että "offset"-arvo on nolla:

**\$0342 D0 00 BNE \$0344**

## Pistekoodit

Pistekoodit ovat eräänlainen suora käskymuoto, jolla voit käynnistää muita toimintoja assemblerin sisällä. Käytössäsi on kuusi pistekoodia: .ASM, .DIS, .MON, .SAV, .LOA ja .GO. Niistä ei välttämättä tarvitse kirjoittaa kuin piste ja ensimmäinen kirjain, MA tunnistaa ne jo siitä.

Jos ohjelmointitaitosi riittävät voit itse lisätä tarpeelliseksi katsomiasi toimintoja lisäämällä rivien 386 ja 410 väleihin samanrakenteisia käskyriivejä, jotka tarkkailevat puolestaan itse määrittelemiäsi pistekoodeja. Lisää Basic-ohjelman perään aliohjelmana haluamasi toiminnon ja keksit sille pistekoodin. Tämän pistekoodin sijoitat esimerkiksi riville 400 samanlaisena kuin edelläkin olevat pistekoodit ja sen perään aliohjelmakutsun omaan rutiiniisi ja sen perään GOTO 330, joka palauttaa tilanteen käskyriivien syöttöön.

## .ASM

Ohjelmoidessasi ilmestyy seuraavan rivin alkuun aina uusi osoite, joka kertoo,

mihin osoitteeseen seuraava käskysi sijoitetaan. Se toimii siis ohjelmalaskurina. Ohjelmalaskuri etenee automaattisesti eteenpäin riippuen siitä, kuinka monta tavua antamasi käsky muistista ottaa.

Voit katkaista tämän tasaisen etenemisen ja siirtyä mihin tahansa muistipaikkaan .ASM käskyllä. Käskyä seuraa yksi välilyönti ja haluttu osoite joko heksana tai desimaalisena.

Kätevimmillään .ASM on korjattaessa eteenpäin tapahtuvien hyppyjen osoitteita jälkikäteen. Sen avulla voit myös jättää ohjelmaan aukon, johon myöhemmin monitorilla sijoitat lukuja, esimerkiksi siinä ohjelmakohdassa tarvittavia, kuva-ruudulle tulostettavia tekstejä. Yleensä tällaiset "datat" sijoitetaan joko ennen ohjelmaa tai ohjelman jälkeen, ei koskaan keskelle. Jos jostain syystä kuitenkin haluat jättää ohjelmaasi aukon, pidä huoli siitä, että viimeinen käsky ennen aukko-kohtaa on hyppykäsky jonnekin muualle (JMP, RTS tms.)

## **.DIS**

Ilman disassembleria olisi tämän kaltainen, koodit suoraan muistiin sijoittava assembler-ohjelma puutteellinen: kun kirjoittamasi käskysanat rullaavat kuva-ruudun ylä laidasta pois, ne häviävät kokonaan. Ainoa muisto niistä on muistipaikkoihin sijoitetut käskykoodit.

Disassembler pelastaa kuitenkin tilanteen. Antamalla käskyn:

**\$036A .DIS \$0342**

LISTaa MA ohjelmaasi osoitteesta \$0342 lähtien tulkiten muistipaikoissa olevia käskykoodeja ja ilmaisten ne kuvaruudulle samanlaisina mnemonisina käskyinä kuin ne olet koneelle antanutkin. Voit ottaa listauksen myös kirjoittimellesi, jos sellainen on käytössäsi, vastaamalla disassemblerin esittämään kysymyksen. .DIS toimii aivan samalla tavalla kuin Basicin LIST, jolla saat ruudulle näkyviin ne ohjelmarivit, jotka ovat jo menneet "piiloon".

Disassembler on hyödyllinen myös, jos haluat tutkia koneesi käyttöjärjestelmän tai Basic-tulkin rakennetta. Tulostuksen saat pysähtymään ja taas käynnistymään painamalla S-näppäintä. Assembleriin voit palata painamalla X:ää.

## **.MON**

Monitorin avulla voit tutkia yksittäisiä muistipaikkoja ja muuttaa niiden sisältöä. Käskyllä

**\$036A .MON \$0342**

saat ruudullesi seuraavanlaisen näytön:

**\$0342 \$D0/**

joka kertoo, että osoitteessa \$0342 on heksaluku D0 (se oli hyppykäskymme koodi edellä). Voit muuttaa muistipaikan sisällön kirjoittamalla /-merkin perään luvun

joko desimaalisena tai heksana (muista \$-merkki). Voit myös säilyttää muistipaikan sisällön entisellään ja siirtyä seuraavaan muistipaikkaan painamalla pelkästään RETURN-näppäintä.

Muut käytössäsi olevat mahdollisuudet ovat:

- (nuoli ylöspäin) = siirrytään edelliseen muistipaikkaan
- (nuoli vasemmalle) = siirrytään takaisin assembleriin.

Monitorin avulla voit sijoittaa varaamillesi data-alueille halutut luvut, esimerkiksi kuvaruudulle kirjoitettavat merkkijonot. Oikeat lukuarvot löydät tällöin mikrosi kuvaruutukoodiluettelosta. Itse ohjelmassa on silloin osuus, joka lataa tältä varaamaltasi alueelta luvut akkuun ja siirtää ne kuvaruutumuistiin. Jos käytät tulostuksessa käyttöjärjestelmän tulostusrutiinia, on koodien oltava ASCII-muodossa.

## .SAV

Ohjelmat on myös saatava talteen kasetille tai levykkeelle. Sen voisit tehdä myös Basicista käsin, mutta konekielisten ohjelmien tallennus vaatii (C-64:ssä) runsasta POKE-käskyjen käyttöä.

Voit tallentaa ohjelman kasetille kirjoittamalla pelkästään .SAV ilman nimeä. Ohjelma taltioituu automaattisesti levyille, jos annat sille nimen käskyn perään, esimerkiksi .SAV OHJELMA1. Jos levyllä on jo tiedosto nimeltä OHJELMA1, kirjoitetaan muistissa oleva ohjelma sen päälle. Ohjelman uudet versiot voi siis tallentaa samalla nimellä tarvitsematta ensin tuhota alkuperäistä.

MA kysyy tallennettavan ohjelman alkuosoitteen ja loppuosoitteen. Voit antaa ne desimaalisina tai heksaoina. Jos vastaat pelkillä RETURNin painalluksilla, sijoittaa MA alkuosoitteeksi sen osoitteen, jonka annoit käynnistäessäsi MA:n ja vastatessasi kysymykseen "Alkuosoite?". Loppuosoitteeksi tulkitaan viimeisin osoite, jossa olet assemblerillasi "käynyt". Disassemblerin ja monitorin käyttö ei vaikuta tähän loppuosoitteeseen.

## .LOA

Tallentamasi konekieliset ohjelmat voit ladata uudestaan muistiin komennolla .LOA. Lataus tapahtuu kasetilta, mikäli komentoa ei seuraa ohjelman nimeä ja levykkeeltä, mikäli komennon perään kirjoitetaan jonkin tiedoston nimi. Konekielinen ohjelma latautuu samasta osoitteesta alkaen, josta se on tallennettukin, ja "ohjelmanalaskuri" käskyrivin vasemmassa laidassa jää osoittamaan ladatun ohjelman viimeistä tavua seuraavaan muistipaikkaan, joten voit samantien jatkaa kesken jääneen ohjelman rakentamista.

## .GO

Tällä komennolla voit käynnistää tekemäsi konekielisen ohjelman. Jos kirjoi-



tat pelkän .GO -komennon, aloitetaan ohjelman suoritus "sisäänkirjoittautumis-osoitteestasi". Voit käynnistää ohjelmasi kuitenkin mistä kohdasta tahansa kirjoittamalla tämän osoitteen komennon perään (muista välilyönti). Esimerkiksi

**\$3400 .GO \$3000**

## **Omien pistekoodien lisääminen**

Kätevä lisäys ohjelmaan olisi esimerkiksi .BYT-komento, jolla voisi syöttää tekstiä näppäimistöltä suoraan peräkkäisiin muistipaikkoihin alueelle, jonka on varannut kuvaaruudulle syötettäville teksteille. Silloin voisi antaa esimerkiksi seuraavat komentorivit:

**\$036E .ASM \$0370**

**\$0370 .BYT SYNTAX ERROR**

ja SYNTAX ERROR-tekstin muodostava ASCII-koodijono olisi jo muistissa. Monitorilla voi tarkistaa lukujen todella olevan muistissa.

Itse asiassa seuraava listaus toteuttaa haaveilemamme .BYT-komennon. Rivit vain lisätään MikroASSEMBLER-ohjelmaan. Applessa on POKE-käskey korvattava ohjelmassa käytetyllä aliohjelmakutsulla.

**396 IF PI\$=".B" THEN 3500**

**3500 GOSUB 4960: IFI=L THEN 3000**

**3510 FOR S=1 TO LEN(N\$); POKE PC,ASC(MID\$(N\$,S,1))**

**3520 PC=PC+1: NEXT S: PRINT: GOTO 345**

Muistiin tulostuva koodijono on ASCII-muodossa, joten sen siirtämiseen kuvaaruudulle kannattaa käyttää käyttöjärjestelmän CHROUT-tyyppistä valmisrutiinia. Koodijono hämmentää disassemblerin, se yrittää vääntää niistä järkeenkäypä konekielikäskyjä. Oikeiden käskyjen sitten alkaessa se ei ymmärrä niistä ensimmäisiä, joten älä disassemblaa koodijonoja. ASCII-jonoksi kelpaa 64:n MikroASSEMBLERISSA kaikki koodiavot 32:sta ylöspäin. Jos tarvitset pienempiä lukuja koodijonoosi, on ne sijoitettava muistiin monitorilla.

## **Muista tallennus!**

Voit olla melko varma siitä, että ohjelmasi ei toimi ensi yrittämällä. Aina on jokin bitti vinossa. Basicissa tämä ilmenee SYNTAX ERRORina ja muina virheilmoituksina. Konekielessä on toisin. Konekielessä ohitat Basic-tulkin ja otat koko koneen komentoosi. Mitään virheilmoituksia ei ole odotettavissa.

Kone noudattaa sokeasti koodiesi viitoittamaa tietä ja jos siellä on virheitä on tuloksena mitä todennäköisimmin koneesi "kaatuminen". Prosessori jää joko hölmöilemään päättymättömään silmukkaan tai sitten se törmää BRK-käskyä vastaavaan koodiin \$00 ja siirtyy sitä kautta esimerkiksi takaisin Basic-tulkin

alaisuuteen ja ruudulle ilmestyy READY.

Joka tapauksessa tuloksena saattaa olla, että kone ei enää tottele komentojasi, ruudulla saattaa vain näkyä prosessorin harhailun aiheuttamaa kajoa. Ainoa keino selvittää tilanteesta on silloin virrat-pois, virrat-päälle.

Ei siis ole mitään syytä käynnistää konekielistä ohjelmaa, ennen kuin se on tallennettu kasetille tai levykkeelle. Silloin ainoaksi harmiksi jää uudelleen lataaminen. Jos tallennus malttamattomuudessa unohtui, voi monen tunnin uurasuksen tulos kutistua hiipuvien elektronien vaimenevaan värähtelyyn.

# MikroASSEMBLER

## COMMODORE 64

Ennen kuin syötät MikroASSEMBLER-listauksen koneeseesi, tutustu ohjelmien syöttöohjeisiin ja Tarkastajan käyttöön kirjan lopusta.

```

10 REM *** MIKROASSEMBLER CBM-64 ***:REM CA
11 REM ***      (C) EEP 1985      ***:REM 87
20 T$="0123456789ABCDEF": GOSUB 6095: GOTO 130:REM 92
30 REM DESHEX JA HEXDES -MUUNNOKSET:REM 9A
40 DE=0:PO=1:FOR M=LEN(HE$) TO 1 STEP-1: IF M=0 THEN
  RETURN:REM 2D
50 FOR G=0 TO 15: IF MID$(HE$,M,1)=MID$(T$,G+1,1) THE
  N 70:REM 6E
60 NEXT G:REM EC
70 DE=DE+PO*G:PO=PO*16:NEXT M: RETURN:REM 17
80 P=2:P%=16: GOTO 100:REM 7E
90 P=4:P%=4096:REM E9
100 HE$="": FOR M=1 TO P: S%=DE/P%:REM AE
110 HE$=HE$+MID$(T$,S%+1,1):DE=DE-S%*P%:P%=P%/16: NEX
  T M: RETURN:REM 24
120 REM TAULUKOIDEN ALUSTUS:REM ED
130 DIM OP$(256),O%(256):REM AD
140 FOR I=1 TO 256: OP$(I)="???":O%(I)=4: NEXT I: FOR
  I=1 TO 38:REM A9
150 READ AS$,P:OP$(P+1)=AS$:O%(P+1)=11+7*(I>8)+2*(I>3
  3)-10*(I>35)-(I>36): NEXT I:REM A8
160 FOR I=1 TO 7: READ AS$,P: FOR J=1 TO 10:REM 76
170 P1=P-4*(J=2)+4*(J=3)+8*(J=6)-8*(J=7)-12*(J=8)-20*
  (J=9)-16*(J=10)+1:REM EF
175 OP$(P1)=AS$: O%(P1)=J: IF J=3 THEN J=5:REM 9C
180 NEXT J:NEXT I: FOR I=1 TO 14: READ AS$:FOR J=1 TO
  10: READ P:IF P=0 THEN 190:REM A4
185 OP$(P+1)=AS$: O%(P+1)=J:REM 85
190 NEXT J: NEXT I: PRINT"<16LEFT><RVS ON>**PAINA NAP
  PIA**":REM D7
200 GET Z$: IF Z$="" THEN 200:REM 79
260 PRINT"<CLR>":REM FC
280 INPUT" ALKUOSOITE";HE$: GOSUB 860: AO=DE: PC=AO:R
  EM DC

```

```

320 PRINT "<CLR>":REM F9
330 PRINT:PRINT" *** ASSEMBLY ***":PRINT:REM 51
345 IN$="":B=0:DE=PC:HE=0:Z=0: GOSUB 90: PRINT "$"HE$
;SPC(10): POKE204,0:REM BA
360 GET K$: IF K$="" THEN 360:REM 69
365 IF K$=CHR$(13) THEN 380:REM 59
367 IFK$=CHR$(20) AND IN$<>"" THEN IN$=LEFT$(IN$,LEN(
IN$)-1):PRINT K$:GOTO360:REM 4F
368 IF ASC(K$)<32 THEN 360:REM 30
370 PRINT K$:IN$=IN$+K$: GOTO 360:REM 64
380 L=LEN(IN$):IF L<2 THEN PRINT:GOTO345:REM 5D
382 POKE 204,1: PRINT" ";: PI$=LEFT$(IN$,2): IF PI$="
.A" THEN 4700:REM 62
386 IF PI$=".D" THEN 4000:REM D3
388 IF PI$=".M" THEN GOSUB 4400: GOTO330:REM 6B
390 IF PI$=".S" THEN GOSUB 4800: GOTO345:REM 74
392 IF PI$=".L" THEN GOSUB 4900: GOTO345:REM 70
394 IF PI$=".G" THEN 4600:REM DB
396 IF PI$=".B" THEN 3500:REM D6
410 REM ** OSOITUSMUODON PÄÄTTELY **:REM C1
420 IF L<3 THEN 3010:REM D3
430 AS$=LEFT$(IN$,3):REM 46
440 IF L=3 THEN OM=4: GOTO 600:REM 28
450 I=4: GOSUB 475: IF L=4 OR I$<>" " THEN 3010:REM F
2
470 I=5: GOSUB 475: IF I$<>"#" THEN 480:REM 92
471 I=I+1: OM=1: IF L<I THEN 3000:REM 97
472 GOSUB 475: IF I$="$" THEN HE=1: I=I+1:REM 27
473 GOSUB 1000: IF F=0 OR F>2ANDHE=1 OR F>3 OR L>=I T
HEN 3000:REM 7B
474 GOTO 600:REM 6E
475 I$=MID$(IN$,I,1): RETURN:REM 40
480 IF I$="$" THEN HE=1: I=I+1:REM CC
485 IF I$="(" THEN I=I+1: GOTO 550:REM AD
490 GOSUB 1000: IF F=0 THEN 3000:REM 4C
495 IF AR>255 THEN Z=1:REM 95
500 IF L<I THEN OM=3-Z: GOTO 600:REM C0
510 IF L-I<1 THEN GOTO 3000:REM 7F
520 IF MID$(IN$,I,2)=","X" THEN OM=8+Z: GOTO 600:REM D
6
530 IF MID$(IN$,I,2)=","Y" THEN OM=13-3*Z: GOTO 600:RE
M 63

```

```

540 GOTO 3000:REM 95
550 IF MID$(IN$,I,1)="$" THEN HE=1:I=I+1:REM 39
560 GOSUB 1000: IF F=0 OR F>5 THEN 3000:REM A4
570 IF L=I+2 THEN 580:REM 26
575 IF MID$(IN$,I,1)=")" THEN OM=12: GOTO 600:REM 2A
580 D$=MID$(IN$,I,3): OM=-6*(D$=",X")-7*(D$="),Y"):
  IF OM=0 THEN 3000:REM E1
600 IF (OM=2 OR OM=3) AND LEFT$(AS$,1)="B" AND AS$(">
  BIT" THEN OM=11:REM 0B
610 FOR I=1 TO 256: IF AS$=OP$(I) THEN IF O%(I)=OM TH
  EN 630:REM FD
620 NEXT I: GOTO 3000:REM 56
630 OC=I-1:REM 0F
790 REM ** TULOSTUS **:REM BF
800 X=1: FOR I=1 TO L+4: PRINT"<LEFT>";:NEXT I: IF OM
  =11 THEN 900:REM F5
810 ON OM GOTO 840,820,840,850,850,840,840,840,820,82
  0,840,820,840:REM 01
820 X=X+1: Y%=AR/256: AR=AR-Y%*256: POKE PC+2,Y%:DE=Y
  %:GOSUB 80:PRINT HE$"<2LEFT>";:REM 1A
840 X=X+1: POKE PC+1,AR: DE=AR: GOSUB 80: PRINT "<2LE
  FT>"HE$;:REM E2
850 POKE PC,OC: DE=OC: GOSUB 80: PRINT "<5LEFT>";HE$:
  PC=PC+X: GOTO 345:REM 08
860 IF LEFT$(HE$,1)="$" THEN HE$=RIGHT$(HE$,LEN(HE$)-
  1):GOSUB 40: RETURN:REM E4
870 DE=VAL(HE$):RETURN:REM 64
900 T=AR-PC-2:IF T<-128 OR T>127 THEN PRINTTAB(24)"LI
  IAN PITKA HYPPY!":GOTO345:REM 22
910 IF T<0 THEN T=T+256:REM C5
920 AR=T: GOTO 810:REM CB
1000 Y=L-I:X=4-HE:FOR F=0TO-X*(X<=Y)-Y*(Y<X): IF L<I
  THEN 1035:REM F7
1010 E=ASC(MID$(IN$,I+F,1)):REM B9
1020 IF E>47AND E<58 OR E>64AND(E<71ANDHE) THEN NEXT:R
  EM 6A
1030 IF HE=1ANDF<>2ANDF<>4 THEN F=0:REM BC
1035 IF F=0 THEN RETURN:REM 1A
1040 HE$=MID$(IN$,I-HE,F+HE):I=I+F: GOSUB 860: AR=DE:
  RETURN:REM 95
3000 PRINT TAB(24)"** VIRHE! ***": GOTO 345:REM A2
3010 PRINT TAB(24)"** VIRHE! MNE": GOTO 345:REM 05

```

```

3020 PRINT TAB(24)"** VIRHE! OSM": GOTO 345:REM 15
3500 GOSUB 4960: IF I=L THEN 3000:REM A8
3510 FOR S=1 TO LEN(N$): POKE PC,ASC(MID$(N$,S,1)):RE
M A3
3520 PC=PC+1: NEXT S: PRINT: GOTO 345:REM 2B
3999 REM:REM C2
4000 PRINT:PRINT " *** DISASSEMBLY ***":REM 98
4005 GOSUB 4960: IF I=L THEN DC=A0: GOTO 4007:REM 78
4006 HE$=N$: GOSUB 860: DC=DE:REM 09
4007 PRINT"RUUDULLE (R) VAI KIRJOITTIMELLE (K)":REM 5
5
4008 INPUT F$:PRINT:REM 8D
4009 IF F$="R" THEN 4015:REM 92
4010 CLOSE1: OPEN1,4: CMD1:REM A8
4015 DE=DC:GOSUB 90: PRINT "$";HE$;" ";:REM 0F
4020 OC=PEEK(DC): AS$=OP$(OC+1): OM=O%(OC+1):REM 89
4025 DE=OC: GOSUB 80: OC$=HE$: IF OM=11 THEN 4340:REM
21
4150 IF OM=4 THEN X=1: GOTO 4190:REM 9C
4160 DE=PEEK(DC+1):GOSUB 80:X$=HE$:DE=PEEK(DC+2):GOSU
B 80:Y$=HE$:REM D1
4170 OC$=OC$+" "+X$: X=2:REM 8C
4180 ON OM GOSUB4210,4220,4230,4230,4230,4260,4270,42
80,4290,4300,4310,4320,4330:REM D8
4190 PRINT OC$;SPC(11-3*X):AS$: DC=DC+X:REM C2
4193 GET A$: IF A$="" THEN 4009:REM C1
4194 IF A$="X" THEN CLOSE1:GOTO 330:REM 7E
4198 GET A$: IF A$="" THEN 4198:REM CF
4200 GOTO 4009:REM CC
4210 AS$=AS$+" #"+X$: RETURN:REM EB
4220 OC$=OC$+Y$+" ": AS$=AS$+" $"+Y$+X$: X=3: RETURN:
REM 6D
4230 AS$=AS$+" $"+X$: RETURN:REM CA
4260 AS$=AS$+" ($"+X$+",X)": RETURN:REM 11
4270 AS$=AS$+" ($"+X$+"),Y": RETURN:REM 13
4280 AS$=AS$+" $"+X$+",X": RETURN:REM C2
4290 GOSUB 4220: AS$=AS$+",X": RETURN:REM 0B
4300 GOSUB 4220: AS$=AS$+",Y": RETURN:REM 04
4320 OC$=OC$+Y$+" ": AS$=AS$+" ($"+Y$+X$+)"": X=3: RE
TURN:REM 2E
4330 AS$=AS$+" $"+X$: RETURN:REM CB

```

```

4340 OF=PEEK(DC+1): DE=OF: GOSUB 80: OC$=OC$+" "+HE$
    REM DB
4350 DE=DC+OF+2: IFOF>127 THEN DE=DE-256:REM B4
4355 GOSUB 90: AS$=AS$+" "$"+HE$:REM 10
4360 X=2: GOTO4190:REM D5
4399 REM:REM BD
4400 PRINT "    *** MONITORI ***":PRINT:REM CD
4410 GOSUB 4960: IF I=L THEN 3000:REM A9
4415 HE$=N$: GOSUB 860: DC=DE:REM 0D
4420 POKE 204,1: PRINT " "; PRINT: DE=DC: GOSUB 90: P
    RINT"$";HE$:REM F9
4425 DE=PEEK(DC): GOSUB 80: PRINT " "$;HE$;"/":REM B8
4430 IN$="":REM 07
4440 POKE 207,0: POKE204,0: GET A$: IF A$="" THEN 444
    0:REM 74
4445 IF A$="+" THEN RETURN:REM B4
4450 IF A$=CHR$(13) THEN 4510:REM AD
4460 IF A$="↑" THEN DC=DC-1: GOTO 4420:REM B6
4470 IF A$=CHR$(20) THEN IN$=LEFT$(IN$,LEN(IN$)-1): P
    RINT A$: GOTO 4440:REM 51
4480 A=ASC(A$): IF A<48AND A>36 OR A>70 OR A>57 AND A
    <65 THEN 4440:REM 4D
4490 IF LEN(IN$)=3 THEN 4440:REM B6
4500 IN$=IN$+A$: PRINT A$: GOTO 4440:REM B2
4510 IF IN$="" THEN 4540:REM 91
4520 IF LEN(IN$)=1 THEN 4440:REM AE
4530 HE$=IN$: GOSUB 860: IF DE<256 THEN POKE DC,DE:RE
    M 92
4540 DC=DC+1: GOTO 4420:REM B1
4598 REM:REM BE
4599 REM ** .GO **:REM 2B
4600 GOSUB 4960: IF I=L THEN SYS A0: GOTO 330:REM 7F
4605 I=I+1: GOSUB 475: IF I$="$" THEN I=I+1: HE=1:REM
    BE
4610 GOSUB 1000: IF F=0 THEN 3000:REM 7A
4620 SYS AR: GOTO 330:REM 67
4638 REM:REM BF
4639 REM ** .ASM **:REM 77
4700 GOSUB 4960: IF I=L THEN 3000:REM AB
4720 I=I+1:GOSUB 475: IF I$="$" THEN I=I+1: HE=1:REM
    BC

```

```

4725 GOSUB 1000: IF F=0 THEN 3000:REM 81
4730 PC=AR: PRINT: PRINT: GOTO 345:REM CE
4798 REM:REM C0
4799 REM ** SAVE **:REM 98
4800 PRINT: SA=AO: INPUT"ALKUOSOITE:"; HE$: IF HE$<>"
  " THEN GOSUB 860: SA=DE:REM 41
4802 SL=PC: INPUT"LOPPUOSOITE:"; HE$: IF HE$<>" THEN
  GOSUB 860: SL=DE+1:REM 54
4805 H%=SL/256:L%=SL-H%*256:A%=SA/256:B%=SA-A%*256:RE
  M D7
4810 GOSUB 4960: IF I=L THEN OPEN 1,1,1: GOTO 4860:RE
  M 4C
4820 OPEN 15,8,15,"I0": OPEN 3,8,1,"0:"+N$+",P,W":REM
  8A
4830 INPUT#15,VN,V$,VT,V: IF VN=0 THEN 4860:REM 91
4840 IF VN<>63 THEN PRINT"**EI ONNISTU**": GOTO 4880:
  REM 93
4850 PRINT#15,"S0:"+N$+",P,W": CLOSE15: CLOSE3: GOTO
  4820:REM 59
4860 POKE157,128: POKE251,B%: POKE252,A%:REM 8D
4870 POKE780,251: POKE781,L%: POKE782,H%: SYS65496:RE
  M F5
4880 CLOSE1: CLOSE3: CLOSE15: PRINT: RETURN:REM 55
4898 REM:REM C1
4899 REM ** LOAD **:REM 8A
4900 GOSUB 4960: IF I=L THEN OPEN 1,1,0: GOTO 4930:RE
  M 49
4910 OPEN 15,8,15,"I0": OPEN 3,8,0,"0:"+N$+",P,R":REM
  84
4920 INPUT#15,VN,V$,VT,V: IF VN THEN PRINT"**EI LÖYDY
  **": GOTO 4880:REM 3E
4930 POKE157,128:POKE185,1:POKE780,0:SYS65493:PC=256*
  PEEK(782)+PEEK(781):PH=PC:REM 3A
4940 IF (ST AND 48) THEN PRINT"**LATAUSVIRHE**":REM 8
  7
4950 GOTO 4880:REM DF
4960 FOR I=1 TO L-1: GOSUB 475: IF I$=" " THEN 4980:R
  EM D3
4970 NEXT I:REM 5C
4980 N$=RIGHT$(IN$,L-I): RETURN:REM 3A

```



```

5000 DATA BCC,144,BCS,176,BEQ,240,BMI,48,BNE,208,BPL,
    16,BVC,80,BVS,112,BRK,0:REM D9
5010 DATA CLC,24,CLD,216,CLI,88,CLV,184,DEX,202,DEY,1
    36,INX,232,INY,200,NOP,234:REM B9
5020 DATA PHA,72,PHP,8,PLA,104,PLP,40,RTI,64,RTS,96,S
    EC,56,SED,248,SEI,120:REM E6
5030 DATA TAX,170,TAY,168,TSX,186,TXA,138,TXS,154,TYA
    ,152,JSR,32,JMP,76,JMP,108:REM 4E
5040 DATA LDX,182,STX,150:REM 7F
5050 DATA ADC,105,AND,41,CMP,201,EOR,73,LDA,169,ORA,9
    ,SBC,233:REM 75
5060 DATA STA,0,141,133,0,0,129,145,149,157,153:REM 4
    C
5070 DATA ASL,0,14,6,10,0,0,0,22,30,0:REM 31
5080 DATA BIT,0,44,36,0,0,0,0,0,0,0:REM CF
5090 DATA CPX,224,236,228,0,0,0,0,0,0,0:REM AA
5100 DATA CPY,192,204,196,0,0,0,0,0,0,0:REM A6
5110 DATA DEC,0,206,198,0,0,0,0,214,222,0:REM EC
5120 DATA INC,0,238,230,0,0,0,0,246,254,0:REM FD
5130 DATA LDX,162,174,166,0,0,0,0,0,0,190:REM 0F
5140 DATA LDY,160,172,164,0,0,0,0,180,188,0:REM 7B
5150 DATA LSR,0,78,70,74,0,0,0,86,94,0:REM 9A
5160 DATA ROL,0,46,38,42,0,0,0,54,62,0:REM 87
5170 DATA ROR,0,110,102,106,0,0,0,118,126,0:REM 71
5180 DATA STX,0,142,134,0,0,0,0,0,0,0:REM 4E
5190 DATA STY,0,140,132,0,0,0,0,148,0,0:REM B9
6095 REM ** ALOITUSRUUTU **:REM 26
6100 PRINT"<CLR>":PRINT" ***** MIKROASSEMBLER *****
    ":REM FA
6120 PRINTSPC(8)"C. EEP 1985 MB":PRINT:REM 81
6130 PRINT" MIKROASSEMBLER SIJOITTAA ANTAMAASI MNEMO
    NISTA KÄSKYÄ VASTAAVAT ":REM CB
6140 PRINT"KOODIT SUORAAN MUISTIIN. MUUT TOIMINNOT VO
    IT KÄYNNISTÄÄ ANTAMALLA ":REM 74
6150 PRINT"RIVIN ALUSSA PISTEKOODIN. PISTE JA ENSIMMÄ
    INEN KIRJAIN RIITTAVAT.":REM E5
6170 PRINT:PRINT" PISTEKOODIT:":REM EF
6180 PRINT"<4SPACE>.ASM ALOITA UUDESTA OSOITTEESTA":
    REM D8
6190 PRINT"<4SPACE>.DIS DISASSEMBLER/KIRJOITINLISTAU
    S":REM 34

```

```
6200 PRINT"<4SPACE>.MON TUTKI JA MUUTA MUISTIPAIKKOJ
A":REM 7E
6210 PRINT"<4SPACE>.SAV TALLETA NAUHALLE/LEVYLLE":RE
M 4F
6220 PRINT"<4SPACE>.LOA LATAA KOODI NAUHALTA/LEVYLTA
":REM 36
6225 PRINT"<4SPACE>.GO KÄYNNISTÄ KONEK. OHJELMA":RE
M F0
6230 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:REM 3B
6240 PRINT"MIKROBITTI<15SPACE>ODOTA HETKI ";; RETURN:
REM DC
```

# MikroASSEMBLER

## VIC-20 +8kt

Ennen kuin syötät MikroASSEMBLER-listauksen koneeseesi, tutustu ohjelmien syöttöohjeisiin ja Tarkastajan käyttöön kirjan lopusta.

MA:n koko on noin 7 kilotavua ja tarvittavat taulukot vievät vielä jonkin verran RAMia. Jos koneessasi on vain 8 kilotavua lisämuistia, voit jättää kaikki REM-lauseet syöttämättä ja näin säästää vapaata tilaa omille konekielisille ohjelmillesi.

```

10 REM *** MIKROASSEMBLER ***
11 REM *** (C) EEP 1985 ***
20 T$="0123456789ABCDEF": GOSUB 6095: GOTO 130:REM 92
30 REM DESHEX JA HEXDES -MUUNNOKSET:REM 9A
40 DE=0:PO=1:FOR M=LEN(HE$) TO 1 STEP-1: IF M=0 THEN
    RETURN:REM 2D
50 FOR G=0 TO 15: IF MID$(HE$,M,1)=MID$(T$,G+1,1) THE
    N 70:REM 6E
60 NEXT G:REM EC
70 DE=DE+PO*G:PO=PO*16:NEXT M: RETURN:REM 17
80 P=2:P%=16: GOTO 100:REM 7E
90 P=4:P%=4096:REM E9
100 HE$="": FOR M=1 TO P: S%=DE/P%:REM AE
110 HE$=HE$+MID$(T$,S%+1,1):DE=DE-S%*P%:P%=P%/16: NEX
    T M: RETURN:REM 24
120 REM TAULUKOIDEN ALUSTUS:REM ED
130 DIM OP$(256),O$(256):REM AD
140 FOR I=1 TO 256: OP$(I)="???":O$(I)=4: NEXT I: FOR
    I=1 TO 38:REM A9
150 READ AS$,P:OP$(P+1)=AS$:O$(P+1)=11+7*(I>8)+2*(I>3
    3)-10*(I>35)-(I>36): NEXT I:REM A8
160 FOR I=1 TO 7: READ AS$,P: FOR J=1 TO 10:REM 76
170 P1=P-4*(J=2)+4*(J=3)+8*(J=6)-8*(J=7)-12*(J=8)-20*
    (J=9)-16*(J=10)+1:REM EF
175 OP$(P1)=AS$: O$(P1)=J: IF J=3 THEN J=5:REM 9C
180 NEXT J:NEXT I: FOR I=1 TO 14: READ AS$:FOR J=1 TO
    10: READ P:IF P=0 THEN 190:REM A4
185 OP$(P+1)=AS$: O$(P+1)=J:REM 85

```

```

190 NEXT J: NEXT I: PRINT"<16LEFT><RVS ON>**PAINA NAP
    PIA**";:REM D7
200 GET Z$: IF Z$="" THEN 200:REM 79
260 PRINT"<CLR>":REM FC
280 INPUT" ALKUOSOITE";HE$: GOSUB 860: AO=DE: PC=AO:R
    EM DC
320 PRINT "<CLR>":REM F9
330 PRINT:PRINT" *** ASSEMBLY ***":PRINT:REM 51
345 IN$="":B=0:DE=PC:HE=0:Z=0: GOSUB 90: PRINT "$"HE$
    ;SPC(10):REM 5F
360 POKE 207,0: POKE 204,0: GET K$: IF K$="" THEN 360
    :REM 22
365 IF K$=CHR$(13) THEN 380:REM 59
367 IFK$=CHR$(20) AND IN$<>"" THEN IN$=LEFT$(IN$,LEN(
    IN$)-1):PRINT K$;:GOTO360:REM 4F
368 IF ASC(K$)<32 THEN 360:REM 30
370 PRINT K$;:IN$=IN$+K$: GOTO 360:REM 64
380 L=LEN(IN$):IF L<2 THEN PRINT:GOTO345:REM 5D
382 POKE 204,1: PRINT" ";: PI$=LEFT$(IN$,2): IF PI$="
    .A" THEN 4700:REM 62
386 IF PI$=".D" THEN 4000:REM D3
388 IF PI$=".M" THEN GOSUB 4400: GOTO330:REM 6B
390 IF PI$=".S" THEN GOSUB 4800: GOTO345:REM 74
392 IF PI$=".L" THEN GOSUB 4900: GOTO345:REM 70
394 IF PI$=".G" THEN 4600:REM DB
396 IF PI$=".B" THEN 3500:REM D6
410 REM ** OSOITUSMUODON PÄÄTELY **:REM C1
420 IF L<3 THEN 3010:REM D3
430 AS$=LEFT$(IN$,3):REM 46
440 IF L=3 THEN OM=4: GOTO 600:REM 28
450 I=4: GOSUB 475: IF L=4 OR I$<>"" THEN 3010:REM F
    2
470 I=5: GOSUB 475: IF I$<>"#" THEN 480:REM 92
471 I=I+1: OM=1: IF L<I THEN 3000:REM 97
472 GOSUB 475: IF I$="$" THEN HE=1: I=I+1:REM 27
473 GOSUB 1000: IF F=0 OR F>2ANDHE=1 OR F>3 OR L>=I T
    HEN 3000:REM 7B
474 GOTO 600:REM 6E
475 I$=MID$(IN$,I,1): RETURN:REM 40
480 IF I$="$" THEN HE=1: I=I+1:REM CC
485 IF I$="(" THEN I=I+1: GOTO 550:REM AD

```

```

490 GOSUB 1000: IF F=0 THEN 3000:REM 4C
495 IF AR>255 THEN Z=1:REM 95
500 IF L<I THEN OM=3-Z: GOTO 600:REM C0
510 IF L-I<1 THEN GOTO 3000:REM 7F
520 IF MID$(IN$,I,2)=","X" THEN OM=8+Z: GOTO 600:REM D
  6
530 IF MID$(IN$,I,2)=","Y" THEN OM=13-3*Z: GOTO 600:RE
  M 63
540 GOTO 3000:REM 95
550 IF MID$(IN$,I,1)="$" THEN HE=1:I=I+1:REM 39
560 GOSUB 1000: IF F=0 OR F>5 THEN 3000:REM A4
570 IF L=I+2 THEN 580:REM 26
575 IF MID$(IN$,I,1)=")" THEN OM=12: GOTO 600:REM 2A
580 D$=MID$(IN$,I,3): OM=-6*(D$=","X")-7*(D$=","Y"):
  IF OM=0 THEN 3000:REM E1
600 IF (OM=2 OR OM=3) AND LEFT$(AS$,1)="B" AND AS$<>"
  BIT" THEN OM=11:REM 0B
610 FOR I=1 TO 256: IF AS$=OP$(I) THEN IF O%(I)=OM TH
  EN 630:REM FD
620 NEXT I: GOTO 3000:REM 56
630 OC=I-1:REM 0F
790 REM ** TULOSTUS **:REM BF
800 X=1: FOR I=1 TO L+4: PRINT"<LEFT>";:NEXT I: IF OM
  =11 THEN 900:REM F5
810 ON OM GOTO 840,820,840,850,850,840,840,840,820,82
  0,840,820,840:REM 01
820 X=X+1: Y%=AR/256: AR=AR-Y%*256: POKE PC+2,Y%:DE=Y
  %:GOSUB 80:PRINT HE$"<2LEFT>";:REM 1A
840 X=X+1: POKE PC+1,AR: DE=AR: GOSUB 80: PRINT "<2LE
  FT>"HE$;:REM E2
850 POKE PC,OC: DE=OC: GOSUB 80: PRINT "<5LEFT>";HE$:
  PC=PC+X: GOTO 345:REM 08
860 IF LEFT$(HE$,1)="$" THEN HE$=RIGHT$(HE$,LEN(HE$)-
  1):GOSUB 40: RETURN:REM E4
870 DE=VAL(HE$):RETURN:REM 64
900 T=AR-PC-2:IF T<-128 OR T>127 THEN PRINTTAB(24)"LI
  IAN PITKA HYPPY!":GOTO345:REM 22
910 IF T<0 THEN T=T+256:REM C5
920 AR=T: GOTO 810:REM CB
1000 Y=L-I:X=4-HE:FOR F=0TO-X*(X<=Y)-Y*(Y<X): IF L<I
  THEN 1035:REM F7

```

```

1010 E=ASC(MID$(IN$,I+F,1)):REM B9
1020 IF E>47AND E<58 OR E>64AND(E<71ANDHE) THEN NEXT:R
    EM 6A
1030 IF HE=1AND F<>2AND F<>4 THEN F=0:REM BC
1035 IF F=0 THEN RETURN:REM 1A
1040 HE$=MID$(IN$,I-HE,F+HE):I=I+F: GOSUB 860: AR=DE:
    RETURN:REM 95
3000 PRINT TAB(24)"** VIRHE! ***": GOTO 345:REM A2
3010 PRINT TAB(24)"** VIRHE! MNE": GOTO 345:REM 05
3020 PRINT TAB(24)"** VIRHE! OSM": GOTO 345:REM 15
3500 GOSUB 4960: IF I=L THEN 3000:REM A8
3510 FOR S=1 TO LEN(N$): POKE PC,ASC(MID$(N$,S,1)):RE
    M A3
3520 PC=PC+1: NEXT S: PRINT: GOTO 345:REM 2B
3999 REM:REM C2
4000 PRINT:PRINT "    *** DISASSEMBLY ***":REM 98
4005 GOSUB 4960: IF I=L THEN DC=A0: GOTO 4007:REM 78
4006 HE$=N$: GOSUB 860: DC=DE:REM 09
4007 PRINT"RUUDULLE (R) VAI KIRJOITTIMELLE (K)":REM 5
    5
4008 INPUT F$:PRINT:REM 8D
4009 IF F$="R" THEN 4015:REM 92
4010 CLOSE1: OPEN1,4: CMD1:REM A8
4015 DE=DC:GOSUB 90: PRINT "$":HE$: "  ":REM 0F
4020 OC=PEEK(DC): AS$=OP$(OC+1): OM=O$(OC+1):REM 89
4025 DE=OC: GOSUB 80: OC$=HE$: IF OM=11 THEN 4340:REM
    21
4150 IF OM=4 THEN X=1: GOTO 4190:REM 9C
4160 DE=PEEK(DC+1):GOSUB 80:X$=HE$:DE=PEEK(DC+2):GOSU
    B 80:Y$=HE$:REM D1
4170 OC$=OC$+" "+X$: X=2:REM 8C
4180 ON OM GOSUB 4210,4220,4230,4230,4230,4260,4270,42
    80,4290,4300,4310,4320,4330:REM D8
4190 PRINT OC$;SPC(11-3*X);AS$: DC=DC+X:REM C2
4193 GET A$: IF A$="" THEN 4009:REM C1
4194 IF A$="X" THEN CLOSE1:GOTO 330:REM 7E
4198 GET A$: IF A$="" THEN 4198:REM CF
4200 GOTO 4009:REM CC
4210 AS$=AS$+" #"+X$: RETURN:REM EB
4220 OC$=OC$+Y$+" ": AS$=AS$+" $"+Y$+X$: X=3: RETURN:
    REM 6D

```

```

4230 AS$=AS$+" $"+"X$": RETURN:REM CA
4260 AS$=AS$+" ($"+X$+",X)": RETURN:REM 11
4270 AS$=AS$+" ($"+X$+"),Y": RETURN:REM 13
4280 AS$=AS$+" $"+"X$+",X": RETURN:REM C2
4290 GOSUB 4220: AS$=AS$+",X": RETURN:REM 0B
4300 GOSUB 4220: AS$=AS$+",Y": RETURN:REM 04
4320 OC$=OC$+Y$+" ": AS$=AS$+" ($"+Y$+X$+")": X=3: RE
TURN:REM 2E
4330 AS$=AS$+" $"+"X$": RETURN:REM CB
4340 OF=PEEK(DC+1): DE=OF: GOSUB 80: OC$=OC$+" "+HE$:
REM DB
4350 DE=DC+OF+2: IFOF>127 THEN DE=DE-256:REM B4
4355 GOSUB 90: AS$=AS$+" $"+"HE$:REM 10
4360 X=2: GOTO4190:REM D5
4399 REM:REM BD
4400 PRINT " *** MONITORI ***":PRINT:REM CD
4410 GOSUB 4960: IF I=L THEN 3000:REM A9
4415 HE$=N$: GOSUB 860: DC=DE:REM 0D
4420 POKE 204,1: PRINT " ": PRINT: DE=DC: GOSUB 90:
PRINT"$";HE$:REM F9
4425 DE=PEEK(DC): GOSUB 80: PRINT" $";HE$;"/":REM B8
4430 IN$="":REM 07
4440 POKE 207,0: POKE 204,0: GET A$: IF A$="" THEN 44
40:REM 74
4445 IF A$="+" THEN RETURN:REM B4
4450 IF A$=CHR$(13) THEN 4510:REM AD
4460 IF A$="↑" THEN DC=DC-1: GOTO 4420:REM B6
4470 IF A$=CHR$(20) THEN IN$=LEFT$(IN$,LEN(IN$)-1): P
RINT A$;: GOTO 4440:REM 51
4480 A=ASC(A$): IF A<48AND A<>36 OR A>70 OR A>57 AND A
<65 THEN 4440:REM 4D
4490 IF LEN(IN$)=3 THEN 4440:REM B6
4500 IN$=IN$+A$: PRINT A$;: GOTO 4440:REM B2
4510 IF IN$="" THEN 4540:REM 91
4520 IF LEN(IN$)=1 THEN 4440:REM AE
4530 HE$=IN$: GOSUB 860: IF DE<256 THEN POKE DC,DE:RE
M 92
4540 DC=DC+1: GOTO 4420:REM B1
4598 REM:REM BE
4599 REM ** .GO **:REM 2B
4600 GOSUB 4960: IF I=L THEN SYS A0: GOTO 330:REM 7F

```

```

4605 I=I+1: GOSUB 475: IF I$="$" THEN I=I+1: HE=1:REM
    BE
4610 GOSUB 1000: IF F=0 THEN 3000:REM 7A
4620 SYS AR: GOTO 330:REM 67
4698 REM:REM BF
4699 REM ** .ASM **:REM 77
4700 GOSUB 4960: IF I=L THEN 3000:REM AB
4720 I=I+1:GOSUB 475: IF I$="$" THEN I=I+1: HE=1:REM
    BC
4725 GOSUB 1000: IF F=0 THEN 3000:REM 81
4730 PC=AR: PRINT: PRINT: GOTO 345:REM CE
4798 REM:REM C0
4799 REM ** SAVE **:REM 98
4800 PRINT: SA=AO: INPUT"ALKUOSOITE:"; HE$: IF HE$<>"
    " THEN GOSUB 860: SA=DE:REM 41
4802 SL=PC: INPUT"LOPPUOSOITE:"; HE$: IF HE$<>" THEN
    GOSUB 860: SL=DE+1:REM 54
4805 H%=SL/256:L%=SL-H%*256:A%=SA/256:B%=SA-A%*256:RE
    M D7
4810 GOSUB 4960: IF I=L THEN OPEN 1,1,1: GOTO 4860:RE
    M 4C
4820 OPEN 15,8,15,"I0": OPEN 3,8,1,"0:"+"N$+",P,W":REM
    8A
4830 INPUT#15,VN,V$,VT,V: IF VN=0 THEN 4860:REM 91
4840 IF VN<>63 THEN PRINT"**EI ONNISTU**": GOTO 4880:
    REM 93
4850 PRINT#15,"S0:"+"N$+",P,W": CLOSE15: CLOSE3: GOTO
    4820:REM 59
4860 POKE157,128: POKE251,B%: POKE252,A%:REM 8D
4870 POKE780,251: POKE781,L%: POKE782,H%: SYS65496:RE
    M F5
4880 CLOSE1: CLOSE3: CLOSE15: PRINT: RETURN:REM 55
4898 REM:REM C1
4899 REM ** LOAD **:REM 8A
4900 GOSUB 4960: IF I=L THEN OPEN 1,1,0: GOTO 4930:RE
    M 49
4910 OPEN 15,8,15,"I0": OPEN 3,8,0,"0:"+"N$+",P,R":REM
    84
4920 INPUT#15,VN,V$,VT,V: IF VN THEN PRINT"**EI LÖYDY
    **": GOTO 4880:REM 3E

```



```

4930 POKE157,128:POKE185,1:POKE780,0:SYS65493:PC=256*
    PEEK(782)+PEEK(781):PH=PC:REM 3A
4940 IF (ST AND 48) THEN PRINT"**:LATAUSVIRHE**":REM 8
    7
4950 GOTO 4880:REM DF
4960 FOR I=1 TO L-1: GOSUB 475: IF I$=" " THEN 4980:R
    EM D3
4970 NEXT I:REM 5C
4980 N$=RIGHT$(IN$,L-I): RETURN:REM 3A
5000 DATA BCC,144,BCS,176,BEQ,240,BMI,48,BNE,208,BPL,
    16,BVC,80,BVS,112,BRK,0:REM D9
5010 DATA CLC,24,CLD,216,CLI,88,CLV,184,DEX,202,DEY,1
    36,INX,232,INY,200,NOP,234:REM B9
5020 DATA PHA,72,PHP,8,PLA,104,PLP,40,RTI,64,RTS,96,S
    EC,56,SED,248,SEI,120:REM E6
5030 DATA TAX,170,TAY,168,TSX,186,TXA,138,TXS,154,TYA
    ,152,JSR,32,JMP,76,JMP,108:REM 4E
5040 DATA LDX,182,STX,150:REM 7F
5050 DATA ADC,105,AND,41,CMP,201,EOR,73,LDA,169,ORA,9
    ,SBC,233:REM 75
5060 DATA STA,0,141,133,0,0,129,145,149,157,153:REM 4
    C
5070 DATA ASL,0,14,6,10,0,0,0,22,30,0:REM 31
5080 DATA BIT,0,44,36,0,0,0,0,0,0,0:REM CF
5090 DATA CPX,224,236,228,0,0,0,0,0,0,0:REM AA
5100 DATA CPY,192,204,196,0,0,0,0,0,0,0:REM A6
5110 DATA DEC,0,206,198,0,0,0,0,214,222,0:REM EC
5120 DATA INC,0,238,230,0,0,0,0,246,254,0:REM FD
5130 DATA LDX,162,174,166,0,0,0,0,0,0,190:REM 0F
5140 DATA LDY,160,172,164,0,0,0,0,180,188,0:REM 7B
5150 DATA LSR,0,78,70,74,0,0,0,86,84,0:REM 9A
5160 DATA ROL,0,46,38,42,0,0,0,54,62,0:REM 87
5170 DATA ROR,0,110,102,106,0,0,0,118,126,0:REM 71
5180 DATA STX,0,142,134,0,0,0,0,0,0,0:REM 4E
5190 DATA STY,0,140,132,0,0,0,0,148,0,0:REM B9
6095 REM **:ALOITUSRUUTU **:REM 26
6100 PRINT"<CLR>": PRINT" *MIKROASSEMBLER*":REM AA
6120 PRINTSPC(5)"C. EEP 1985 MB":PRINT:PRINT:REM 45
6180 PRINT".ASM ALOITA UUD. OS.":REM 55
6190 PRINT".DIS DISASM/KIRJ.LIST ":REM 39
6200 PRINT".MON TUTKI/MUUTA MUIST":REM 8F

```

```
6210 PRINT".SAV TALLETA NAU/LEV":REM B3
6220 PRINT".LOA LATAA NAU/LEVYLTA":REM 56
6225 PRINT".GO KÄYNNISTÄ OHJELMA":REM 4A
6230 PRINT:PRINT:PRINT:REM E6
6240 PRINT"MIKROBITTI<22SPACE>ODOTA HETKI ";; RETURN:
      REM DC
```

# MikroASSEMBLER

## VIC-20 +3kt

Jos VICissäsi ei ole lisämuistia lainkaan, ei MikroASSEMBLER-ohjelma mahdu Basic-ohjelmille varattuun tilaan edes paloitetuna.

Jos VICissäsi on 3 kilotavun lisämuisti, jää Basic-ohjelmille tilaa noin 6,5 kilotavua. Koko MikroASSEMBLER-ohjelma ei siihen vielä mahdu, mutta jakamalla sen kahteen osaan voit käyttää kaikkia toimintoja silti hyväksesi. Seuraavissa listauksissa MA on jaettu kahteen lohkoon, joista ensimmäisessä ovat assembler, lataus, tallennus ja konekielisten ohjelmien käynnistys ja toisessa lohkossa ovat disassembler ja monitori. Tallenna ne erillisinä ohjelmoina kasetille tai levykkeelle ja lataa niistä aina se, jota tarvitset. Konekielist ohjelmat säilyvät koneen muistissa toisen ohjelman latauksenkin aikana ellet välillä katkaise virtaa tai ellei ohjelma sijaitse kasetilta ladattaessa kasettipuskurin alueella.

```

5 REM = JÄTÄ KAIKKI REM-LAUSEET SYÖTTÄMÄTTÄ =:REM 75
6 REM = JOTTA OHJELMA SOPIISI VICIN MUISTIIN =:REM D8
10 REM *** MIKROASSEMBLER VIC-20 ***:REM D2
20 T$="0123456789ABCDEF": GOSUB 6100: GOTO 130:REM 85
30 REM DESHEX JA HEXDES -MUUNNOKSET:REM 9A
40 DE=0:PO=1:FOR M=LEN(HE$) TO 1 STEP-1: IF M=0 THEN
    RETURN:REM 2D
50 FOR G=0 TO 15: IF MID$(HE$,M,1)=MID$(T$,G+1,1) THEN
    N 70:REM 6E
60 NEXT G:REM EC
70 DE=DE+PO*G:PO=PO*16:NEXT M: RETURN:REM 17
80 P=2:P%=16: GOTO 100:REM 7E
90 P=4:P%=4096:REM E9
100 HE$="": FOR M=1 TO P: S%=DE/P%:REM AE
110 HE$=HE$+MID$(T$,S%+1,1):DE=DE-S%*P%:P%=P%/16: NEX
    T M: RETURN:REM 24
120 REM TAULUKOIDEN ALUSTUS:REM ED
130 DIM OP$(256),O%(256):REM AD
140 FOR I=1 TO 256: OP$(I)="???":O%(I)=4: NEXT I: FOR
    I=1 TO 38:REM A9

```

```

150 READ AS$,P:OP$(P+1)=AS$:O%(P+1)=11+7*(I>8)+2*(I>3
3)-10*(I>35)-(I>36): NEXT I:REM A8
160 FOR I=1 TO 7: READ AS$,P: FOR J=1 TO 10:REM 76
170 P1=P-4*(J=2)+4*(J=3)+8*(J=6)-8*(J=7)-12*(J=8)-20*
(J=9)-16*(J=10)+1:REM EF
175 OP$(P1)=AS$: O%(P1)=J: IF J=3 THEN J=5:REM 9C
180 NEXT J:NEXT I: FOR I=1 TO 14: READ AS$:FOR J=1 TO
10: READ P:IF P=0 THEN 190:REM A4
185 OP$(P+1)=AS$: O%(P+1)=J:REM 85
190 NEXT J: NEXT I:REM E5
260 PRINT"<CLR>":REM FC
280 INPUT" ALKUOSOITE";HE$: GOSUB 860: AO=DE: PC=AO:R
EM DC
320 PRINT "<CLR>":REM F9
330 PRINT:PRINT" *** ASSEMBLY ***":PRINT:REM 51
345 IN$="":B=0:DE=PC:HE=0:Z=0: GOSUB 90: PRINT "$"HE$
;SPC(10):REM 5F
360 POKE 207,0: POKE 204,0: GET K$: IF K$="" THEN 360
:REM 22
365 IF K$=CHR$(13) THEN 380:REM 59
367 IFK$=CHR$(20) AND IN$<>"" THEN IN$=LEFT$(IN$,LEN(
IN$)-1):PRINT K$:GOTO360:REM 4F
368 IF ASC(K$)<32 THEN 360:REM 30
370 PRINT K$;:IN$=IN$+K$: GOTO 360:REM 64
380 L=LEN(IN$):IF L<2 THEN PRINT:GOTO345:REM 5D
382 PRINT" "+CHR$(20);: POKE204,1: P1$=LEFT$(IN$,2):
IF P1$=".A" THEN 4700:REM 41
390 IF P1$=".S" THEN GOSUB 4800: GOTO345:REM 74
392 IF P1$=".L" THEN GOSUB 4900: GOTO345:REM 70
394 IF P1$=".G" THEN 4600:REM DB
396 REM TAHAN LISAKOODIEN TARKISTUKSET:REM 20
410 REM ** OSOITUSMUODON PÄÄTELTY ** :REM C1
420 IF L<3 THEN 3010:REM D3
430 AS$=LEFT$(IN$,3):REM 46
440 IF L=3 THEN OM=4: GOTO 600:REM 28
450 I=4: GOSUB 475: IF L=4 OR I$<>" " THEN 3010:REM F
2
470 I=5: GOSUB 475: IF I$<>"#" THEN 480:REM 92
471 I=I+1: OM=1: IF L<I THEN 3000:REM 97
472 GOSUB 475: IF I$="$" THEN HE=1: I=I+1:REM 27

```

```

473 GOSUB 1000: IF F=0 OR F>2ANDHE=1 OR F>3 OR L>=1 T
HEN 3000:REM 7B
474 GOTO 600:REM 6E
475 I$=MID$(IN$,I,1): RETURN:REM 40
480 IF I$="$" THEN HE=1: I=I+1:REM CC
485 IF I$("<" THEN I=I+1: GOTO 550:REM AD
490 GOSUB 1000: IF F=0 THEN 3000:REM 4C
495 IF AR>255 THEN Z=1:REM 95
500 IF L<I THEN OM=3-Z: GOTO 600:REM C0
510 IF L-I<1 THEN GOTO 3000:REM 7F
520 IF MID$(IN$,I,2)=","X" THEN OM=8+Z: GOTO 600:REM D
6
530 IF MID$(IN$,I,2)=","Y" THEN OM=13-3*Z: GOTO 600:RE
M 63
540 GOTO 3000:REM 95
550 IF MID$(IN$,I,1)="$" THEN HE=1:I=I+1:REM 39
560 GOSUB 1000: IF F=0 OR F>5 THEN 3000:REM A4
570 IF L=I+2 THEN 580:REM 26
575 IF MID$(IN$,I,1)=")" THEN OM=12: GOTO 600:REM 2A
580 D$=MID$(IN$,I,3): OM=-6*(D$=","X")-7*(D$=","Y"):
IF OM=0 THEN 3000:REM E1
600 IF (OM=2 OR OM=3) AND LEFT$(AS$,1)="B" AND AS$("<)"
BIT" THEN OM=11:REM 0B
610 FOR I=1 TO 256: IF AS$=OP$(I) THEN IF O%(I)=OM TH
EN 630:REM FD
620 NEXT I: GOTO 3000:REM 56
630 OC=I-1:REM 0F
790 REM ** TULOSTUS **:REM BF
800 X=1: FOR I=1 TO L+4: PRINT"<LEFT>";:NEXT I: IF OM
=11 THEN 900:REM F5
810 ON OM GOTO 840,820,840,850,850,840,840,840,820,82
0,840,820,840:REM 01
820 X=X+1: Y%=AR/256: AR=AR-Y%*256: POKE PC+2,Y%:DE=Y
%:GOSUB 80:PRINT HE$("<LEFT>");:REM 1A
840 X=X+1: POKE PC+1,AR: DE=AR: GOSUB 80: PRINT "<2LE
FT>"HE$:REM E2
850 POKE PC,OC: DE=OC: GOSUB 80: PRINT "<5LEFT>";HE$:
PC=PC+X: GOTO 345:REM 08
860 IF LEFT$(HE$,1)="$" THEN HE$=RIGHT$(HE$,LEN(HE$)-
1):GOSUB 40: RETURN:REM E4
870 DE=VAL(HE$):RETURN:REM 64

```

```

900 T=AR-PC-2:IF T<-128 OR T>127 THEN PRINTTAB(24)"LI
    IAN PITKÄ HYPPY!":GOTO345:REM 22
910 IF T<0 THEN T=T+256:REM C5
920 AR=T: GOTO 810:REM C8
1000 Y=L-I:X=4-HE:FOR F=0TO-X*(X<=Y)-Y*(Y<X): IF L<I
    THEN 1035:REM F7
1010 E=ASC(MID$(IN$,I+F,1)):REM B9
1020 IF E>47AND E<58 OR E>64AND(E<71ANDHE) THEN NEXT:R
    EM 6A
1030 IF HE=1ANDF<>2ANDF<>4 THEN F=0:REM BC
1035 IF F=0 THEN RETURN:REM 1A
1040 HE$=MID$(IN$,I-HE,F+HE):I=I+F: GOSUB 860: AR=DE:
    RETURN:REM 95
3000 PRINT TAB(24)"** VIRHE! ***": GOTO 345:REM A2
3010 PRINT TAB(24)"** VIRHE! MNE": GOTO 345:REM 05
3020 PRINT TAB(24)"** VIRHE! OSM": GOTO 345:REM 15
4598 REM:REM BE
4599 REM ** .GO **:REM 2B
4600 GOSUB 4960: IF I=L THEN SYS A0: GOTO 330:REM 7F
4605 I=I+1: GOSUB 475: IF I$="$" THEN I=I+1: HE=1:REM
    BE
4610 GOSUB 1000: IF F=0 THEN 3000:REM 7A
4620 SYS AR: GOTO 330:REM 67
4698 REM:REM BF
4699 REM ** .ASM **:REM 77
4700 GOSUB 4960: IF I=L THEN 3000:REM AB
4720 I=I+1:GOSUB 475: IF I$="$" THEN I=I+1: HE=1:REM
    BC
4725 GOSUB 1000: IF F=0 THEN 3000:REM 81
4730 PC=AR: PRINT: PRINT: GOTO 345:REM CE
4798 REM:REM C0
4799 REM ** SAVE **:REM 98
4800 PRINT: SA=A0: INPUT"ALKUOSOITE: "; HE$: IF HE$<>"
    " THEN GOSUB 860: SA=DE:REM 41
4802 SL=PC: INPUT"LOPPUOSOITE: "; HE$: IF HE$<>" " THEN
    GOSUB 860: SL=DE+1:REM 54
4805 H%=SL/256:L%=SL-H%*256:A%=SA/256:B%=SA-A%*256:RE
    M D7
4810 GOSUB 4960: IF I=L THEN OPEN 1,1,1: GOTO 4860:RE
    M 4C

```

```

4820 OPEN 15,8,15,"I0": OPEN 3,8,1,"0:"+N$+",P,W":REM
    8A
4830 INPUT#15,VN,V$,VT,V: IF VN=0 THEN 4860:REM 91
4840 IF VN<>63 THEN PRINT"**EI ONNISTU**": GOTO 4880:
    REM 93
4850 PRINT#15,"S0:"+N$+",P,W": CLOSE15: CLOSE3: GOTO
    4820:REM 59
4860 POKE157,128: POKE251,B%: POKE252,A%:REM 8D
4870 POKE780,251: POKE781,L%: POKE782,H%: SYS65496:RE
    M F5
4880 CLOSE1: CLOSE3: CLOSE15: PRINT: RETURN:REM 55
4898 REM:REM C1
4899 REM ** LOAD **:REM 8A
4900 GOSUB 4960: IF I=L THEN OPEN 1,1,0: GOTO 4930:RE
    M 49
4910 OPEN 15,8,15,"I0": OPEN 3,8,0,"0:"+N$+",P,R":REM
    84
4920 INPUT#15,VN,V$,VT,V: IF VN THEN PRINT"**EI LÖYDY
    **": GOTO 4880:REM 3E
4930 POKE157,128:POKE185,1:POKE780,0:SYS65493:PC=256*
    PEEK(782)+PEEK(781):PH=PC:REM 3A
4940 IF (ST AND 48) THEN PRINT"**LATAUSVIRHE**":REM 8
    7
4950 GOTO 4880:REM DF
4960 FOR I=1 TO L-1: GOSUB 475: IF I$=" " THEN 4980:R
    EM D3
4970 NEXT I:REM 5C
4980 N$=RIGHT$(IN$,L-I): RETURN:REM 3A
5000 DATA BCC,144,BCS,176,BEQ,240,BMI,48,BNE,208,BPL,
    16,BVC,80,BVS,112,BRK,0:REM D9
5010 DATA CLC,24,CLD,216,CLI,88,CLV,184,DEX,202,DEY,1
    36,INX,232,INY,200,NOP,234:REM B9
5020 DATA PHA,72,PHP,8,PLA,104,PLP,40,RTI,64,RTS,96,S
    EC,56,SED,248,SEI,120:REM E6
5030 DATA TAX,170,TAY,168,TSX,186,TXA,138,TXS,154,TYA
    ,152,JSR,32,JMP,76,JMP,108:REM 4E
5040 DATA LDX,182,STX,150:REM 7F
5050 DATA ADC,105,AND,41,CMP,201,EOR,73,LDA,169,ORA,9
    ,SBC,233:REM 75
5060 DATA STA,0,141,133,0,0,129,145,149,157,153:REM 4
    C

```

```

5070 DATA ASL,0,14,6,10,0,0,0,22,30,0:REM 31
5080 DATA BIT,0,44,36,0,0,0,0,0,0,0:REM CF
5090 DATA CPX,224,236,228,0,0,0,0,0,0,0:REM AA
5100 DATA CPY,192,204,196,0,0,0,0,0,0,0:REM A6
5110 DATA DEC,0,206,198,0,0,0,0,214,222,0:REM EC
5120 DATA INC,0,238,230,0,0,0,0,246,254,0:REM FD
5130 DATA LDX,162,174,166,0,0,0,0,0,0,0:REM 0F
5140 DATA LDY,160,172,164,0,0,0,0,180,188,0:REM 7B
5150 DATA LSR,0,78,70,74,0,0,0,86,94,0:REM 9A
5160 DATA ROL,0,46,38,42,0,0,0,54,62,0:REM 87
5170 DATA ROR,0,110,102,106,0,0,0,118,126,0:REM 71
5180 DATA STX,0,142,134,0,0,0,0,0,0,0:REM 4E
5190 DATA STY,0,140,132,0,0,0,0,148,0,0:REM B9
6095 REM ** ALOITUSRUUTU **:REM 26
6100 PRINT"<CLR>": PRINT" *MIKROASSEMBLER* ":REM AA
6230 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:REM 3B
6240 PRINT"<9SPACE>ODOTA HETKI "': RETURN:REM DE

```

## DISASSEMBLER JA MONITORI

```

20 T$="0123456789ABCDEF": GOSUB 6095: GOTO 130:REM 92
30 REM DESHEX JA HEXDES -MUUNNOKSET:REM 9A
40 DE=0:PO=1:FOR M=LEN(HE$) TO 1 STEP-1: IF M=0 THEN
    RETURN:REM 2D
50 FOR G=0 TO 15: IF MID$(HE$,M,1)=MID$(T$,G+1,1) THEN
    N 70:REM 6E
60 NEXT G:REM EC
70 DE=DE+PO*G:PO=PO*16:NEXT M: RETURN:REM 17
80 P=2:P%=16: GOTO 100:REM 7E
90 P=4:P%=4096:REM E9
100 HE$="": FOR M=1 TO P: S%=DE/P%:REM AE
110 HE$=HE$+MID$(T$,S%+1,1):DE=DE-S%*P%:P%=P%/16: NEX
    T M: RETURN:REM 24
120 REM TAULUKOIDEN ALUSTUS:REM ED

```



```

130 DIM OP$(256),O%(256):REM AD
140 FOR I=1 TO 256: OP$(I)="???":O%(I)=4: NEXT I: FOR
  I=1 TO 38:REM A9
150 READ AS$,P:OP$(P+1)=AS$:O%(P+1)=11+7*(I>8)+2*(I>3
  3)-10*(I>35)-(I>36): NEXT I:REM A8
160 FOR I=1 TO 7: READ AS$,P: FOR J=1 TO 10:REM 76
170 P1=P-4*(J=2)+4*(J=3)+8*(J=6)-8*(J=7)-12*(J=8)-20*
  (J=9)-16*(J=10)+1:REM EF
175 OP$(P1)=AS$: O%(P1)=J: IF J=3 THEN J=5:REM 9C
180 NEXT J:NEXT I: FOR I=1 TO 14: READ AS$:FOR J=1 TO
  10: READ P:IF P=0 THEN 190:REM A4
185 OP$(P+1)=AS$: O%(P+1)=J:REM 85
190 NEXT J: NEXT I:REM E5
195 PRINT"<16LEFT><RVS ON>**PAINA NAPPIA**":REM 57
200 GET A$: IF A$="" THEN 200:REM 47
210 IF A$="1" THEN 4000:REM 2C
220 IF A$="2" THEN 4400:REM 32
230 GOTO 200:REM 60
475 I$=MID$(IN$,I,1): RETURN:REM 40
860 IF LEFT$(HE$,1)="$" THEN HE$=RIGHT$(HE$,LEN(HE$)-
  1):GOSUB 40: RETURN:REM E4
870 DE=VAL(HE$):RETURN:REM 64
4000 PRINT:PRINT "<CLR> *** DISASSEMBLY ***":REM 2B
4005 INPUT"ALKUOSOITE":HE$: GOSUB 860: DC=DE:REM 68
4007 PRINT"RUUDULLE (R) VAI KIRJOITTIMELLE (K)":REM 5
  5
4008 INPUT F$:PRINT:REM 8D
4009 IF F$="R" THEN 4015:REM 92
4010 CLOSE1: OPEN1,4: CMD1:REM A8
4015 DE=DC:GOSUB 90: PRINT "$":HE$;" ":REM 0F
4020 OC=PEEK(DC): AS$=OP$(OC+1): OM=O%(OC+1):REM 89
4025 DE=OC: GOSUB 80: OC$=HE$: IF OM=11 THEN 4340:REM
  21
4150 IF OM=4 THEN X=1: GOTO 4190:REM 9C
4160 DE=PEEK(DC+1):GOSUB 80:X$=HE$:DE=PEEK(DC+2):GOSU
  B 80:Y$=HE$:REM D1
4170 OC$=OC$+" "+X$: X=2:REM 8C
4180 ON OM GOSUB4210,4220,4230,4230,4230,4260,4270,42
  80,4290,4300,4310,4320,4330:REM D8
4190 PRINT OC$:SPC(9-3*X):AS$: DC=DC+X:REM 99
4193 GET A$: IF A$="" THEN 4009:REM C1

```

```

4194 IF A$="X" THEN CLOSE1:GOSUB 6100: GOTO 195:REM 0
8
4198 GET A$: IF A$="" THEN 4198:REM CF
4200 GOTO 4009:REM CC
4210 A$=A$+" #"+X$: RETURN:REM EB
4220 OC$=OC$+Y$+" ": A$=A$+" $"+Y$+X$: X=3: RETURN:
REM 6D
4230 A$=A$+" $"+X$: RETURN:REM CA
4260 A$=A$+" ($"+X$+",X)": RETURN:REM 11
4270 A$=A$+" ($"+X$+"),Y": RETURN:REM 13
4280 A$=A$+" $"+X$+",X": RETURN:REM C2
4290 GOSUB 4220: A$=A$+",X": RETURN:REM 0B
4300 GOSUB 4220: A$=A$+",Y": RETURN:REM 04
4320 OC$=OC$+Y$+" ": A$=A$+" ($"+Y$+X$+"): X=3: RE
TURN:REM 2E
4330 A$=A$+" $"+X$: RETURN:REM CB
4340 OF=PEEK(DC+1): DE=OF: GOSUB 80: OC$=OC$+" "+HE$:
REM DB
4350 DE=DC+OF+2: IFOF>127 THEN DE=DE-256:REM B4
4355 GOSUB 90: A$=A$+" $"+HE$:REM 10
4360 X=2: GOTO4190:REM D5
4399 REM:REM BD
4400 PRINT "<CLR> *** MONITORI ***":PRINT:REM 60
4410 INPUT"ALKUOSOITE";HE$:GOSUB 860: DC=DE:REM 68
4420 POKE 204,1: PRINT " ": PRINT: DE=DC: GOSUB 90: P
RINT"$";HE$;:REM F9
4425 DE=PEEK(DC): GOSUB 80: PRINT " $";HE$;"/";:REM B8
4430 IN$="":REM 07
4440 POKE 207,0: POKE 204,0: GET A$: IF A$="" THEN 44
40:REM 74
4445 IF A$="+" THEN GOSUB 6100: GOTO 195:REM 2D
4450 IF A$=CHR$(13) THEN 4510:REM AD
4460 IF A$="↑" THEN DC=DC-1: GOTO 4420:REM B6
4470 IFA$=CHR$(20) AND IN$<>"" THEN IN$=LEFT$(IN$,LEN
(IN$)-1):PRINT A$;:GOTO4440:REM 3D
4480 A=ASC(A$): IF A<48ANDA<>36 OR A>70 OR A>57 AND A
<65 THEN 4440:REM 4D
4490 IF LEN(IN$)=3 THEN 4440:REM B6
4500 IN$=IN$+A$: PRINT A$;: GOTO 4440:REM B2
4510 IF IN$="" THEN 4540:REM 91
4520 IF LEN(IN$)=1 THEN 4440:REM AE

```

```

4530 HE#=IN#: GOSUB 860: IF DE<256 THEN POKE DC,DE:RE
M 32
4540 DC=DC+1: GOTO 4420:REM B1
5000 DATA BCC,144,BCS,176,BEQ,240,BMI,48,BNE,208,BPL,
16,BVC,80,BVS,112,BRK,0:REM D9
5010 DATA CLC,24,CLD,216,CLI,88,CLV,184,DEX,202,DEY,1
36,INX,232,INY,200,NOP,234:REM B9
5020 DATA PHA,72,PHP,8,PLA,104,PLP,40,RTI,64,RTS,96,S
EC,56,SED,248,SEI,120:REM E6
5030 DATA TAX,170,TAY,168,TSX,186,TXA,138,TXS,154,TYA
,152,JSR,32,JMP,76,JMP,108:REM 4E
5040 DATA LDX,182,STX,150:REM 7F
5050 DATA ADC,105,AND,41,CMP,201,EOR,73,LDA,169,ORA,9
,SBC,233:REM 75
5060 DATA STA,0,141,133,0,0,129,145,149,157,153:REM 4
C
5070 DATA ASL,0,14,6,10,0,0,0,22,30,0:REM 31
5080 DATA BIT,0,44,36,0,0,0,0,0,0,0:REM CF
5090 DATA CPX,224,236,228,0,0,0,0,0,0,0:REM AA
5100 DATA CPY,192,204,196,0,0,0,0,0,0,0:REM A6
5110 DATA DEC,0,206,198,0,0,0,0,214,222,0:REM EC
5120 DATA INC,0,238,230,0,0,0,0,246,254,0:REM FD
5130 DATA LDX,162,174,166,0,0,0,0,0,0,190:REM 0F
5140 DATA LDY,160,172,164,0,0,0,0,180,188,0:REM 7B
5150 DATA LSR,0,78,70,74,0,0,0,86,94,0:REM 9A
5160 DATA ROL,0,46,38,42,0,0,0,54,62,0:REM 87
5170 DATA ROR,0,110,102,106,0,0,0,118,126,0:REM 71
5180 DATA STX,0,142,134,0,0,0,0,0,0,0:REM 4E
5190 DATA STY,0,140,132,0,0,0,0,148,0,0:REM B9
6095 REM ** ALOITUSRUUTU **:REM 26
6100 PRINT"<CLR>": PRINT"*DISASSEMBLERI JA MONITORI*"
:PRINT:PRINT:REM DB
6120 PRINTSPC(4)"C. EEP 1985 MB":PRINT:REM 7D
6130 PRINT SPC(3)"VALITSE TOIMINTO:":REM CA
6140 PRINT:PRINTSPC(6)"1 DISASSEMBLER":REM 7F
6150 PRINT SPC(6)"2 MONITORI":REM AD
6230 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:REM 3B
6240 PRINT"<9SPACE>ODOTA HETKI "": RETURN:REM DE

```

# MikroASSEMBLER

## Apple II

Ennen kuin syötät listauksen koneeseesi, tutustu ohjelmien syöttöohjeisiin kirjan liitteestä.

Applen MikroASSEMBLER toimii pääpiirteiltään kuten aiemmin on selostettu. Kuitenkin eräitä ominaisuuksia on muutettu. Ohjelma on suunniteltu tallentamaan ohjelmia pelkästään levyille, joten kasettioperaatioita ei voi käynnistää. Pistekoodit ovat samat, mutta disassemblerin sekä monitorin ohjausnäppäimiä on muutettu.

Ollessasi monitorissa, siirryt tavun eteenpäin painamalla RETURNia. Taaksepäin pääsee TABilla (Control-I, jollei TAB-näppäintä ole). Monitorista poistutaan painamalla ESC-näppäintä. Disassemblauksesta pääsee pois X-näppäimen painalluksella.

```

1  REM  **** MIKRO ASSEMBLER ****
2  REM
3  REM  converted to Apple II by
4  REM  Aki Korhonen
5  REM
15 H$ = CHR$(8): REM  BACKSPACE
16 D$ = CHR$(4): REM  DOS KOMENTOJA VARTEN
20 T$="0123456789ABCDEF": GOSUB 6095: GOTO 130
30 REM  ** DES <--> HES KÄÄNNÖKSET **
40 DE = 0:PO = 1: FOR M = LEN(HE$) TO 1 STEP
- 1: IF M = 0 THEN RETURN
50 FOR G = 0 TO 15: IF MID$(HE$,M,1) =
MID$(T$,G + 1,1) THEN 70
60 NEXT G
70 DE = DE + PO * G:PO = PO * 16: NEXT M:
RETURN
80 P = 2:P% = 16: GOTO 100
90 P = 4:P% = 4096
100 HE$ = "": FOR M = 1 TO P:S% = DE / P%
110 HE$ = HE$ + MID$(T$,S% + 1,1):DE = DE -
S% * P%:P% = P% / 16: NEXT M: RETURN
120 REM  ** TAULUKOIDEN ALUSTUS **
130 DIM OP$(256),O%(256)
140 FOR I = 1 TO 256:OP$(I) = "???":O%(I) =
4: NEXT I: FOR I = 1 TO 38

```

```

150 READ AS$,P:OP$(P + 1) = AS$:O%(P + 1) =
11 - 7 * (I > 8) - 2 * (I > 33) + 10 * (I > 35)
+ (I > 36):: NEXT
160 FOR I = 1 TO 7: READ AS$,P: FOR J = 1 TO
10
170 P1 = P + 4 * (J = 2) - 4 * (J = 3) - 8 *
(J = 6) + 8 * (J = 7) + 12 * (J = 8) + 20 * (J
= 9) + 16 * (J = 10) + 1
175 OP$(P1)=AS$: O%(P1) = J: IF J=3 THEN J = 5
180 NEXT J: NEXT I: FOR I = 1 TO 14: READ
AS$: FOR J = 1 TO 10: READ P: IF P = 0 THEN 190
185 OP$(P + 1) = AS$:O%(P + 1) = J
190 NEXT J: NEXT I
260 HOME
280 INPUT " ALKUOSOITE >";HE$: GOSUB 860:AO =
DE:PC = AO
320 HOME
330 PRINT : PRINT " ASSEMBLY": PRINT
345 IN$ = "":B = 0:DE = PC:HE = 0:Z = 0: GOSUB
90: PRINT "$";HE$: TAB( 20)
360 GET K$
362 IF ASC (K$) < 32 AND NOT (K$ = H$ OR K$
= CHR$(13)) THEN 360
365 IF K$ = CHR$(13) THEN 380
367 IF K$ = H$ AND LEN (IN$) > 1 THEN IN$ =
LEFT$ (IN$, LEN (IN$) - 1): PRINT H$:: GOTO 360
368 IF K$ = H$ AND LEN (IN$) = 1 THEN IN$ =
"": PRINT H$:: GOTO 360
369 IF K$ = H$ THEN 360
370 PRINT K$::IN$ = IN$ + K$: GOTO 360
380 L = LEN (IN$): IF L < 2 THEN PRINT :
GOTO 345
382 PI$ = LEFT$ (IN$,2): IF PI$ = ".A" THEN
4700
384 IF PI$ = ".D" THEN 4000
386 IF PI$ = ".M" THEN GOSUB 4400: GOTO 330
388 IF PI$ = ".S" THEN GOSUB 4800: GOTO 345
390 IF PI$ = ".L" THEN GOSUB 4900: GOTO 345
392 IF PI$ = ".G" THEN 4600
394 REM ** Tähän voi lisätä uusia komentoja
400 REM
410 REM ** Osoitusmuodon tarkistus **
420 IF L < 3 THEN 3010
430 AS$ = LEFT$ (IN$,3)
440 IF L = 3 THEN OM = 4: GOTO 600
450 I = 4: GOSUB 475: IF L = 4 OR I$ < > " "
THEN 3010
470 I = 5: GOSUB 475: IF I$ < > "#" THEN 480
471 I = I + 1:OM = 1: IF L < I THEN 3000
472 GOSUB 475: IF I$ = "$" THEN HE = 1:I = I
+ 1
473 GOSUB 1000: IF F = 0 OR F > 2 AND HE = 1
OR F > 3 OR L > = I THEN 3000
474 GOTO 600
475 I$ = MID$ (IN$,I,1): RETURN

```

```

480 IF I$ = "$" THEN HE = 1: I = I + 1
485 IF I$ = "(" THEN I = I + 1: GOTO 550
490 GOSUB 1000: IF F = 0 THEN 3000
495 IF AR > 255 THEN Z = 1
500 IF L < I THEN OM = 3 - Z: GOTO 600
510 IF L - I < 1 THEN 3000
520 IF MID$ (IN$, I, 2) = ",X" THEN OM = 8 +
Z: GOTO 600
530 IF MID$ (IN$, I, 2) = ",Y" THEN OM = 13 -
3 * Z: GOTO 600
540 GOTO 3000
550 IF MID$ (IN$, I, 1) = "$" THEN HE = 1: I = I
+ 1
560 GOSUB 1000: IF F = 0 OR F > 5 THEN 3000
570 IF L = I + 2 THEN 590
580 IF MID$ (IN$, I, 1) = ")" THEN OM = 12:
GOTO 600
590 D$ = MID$ (IN$, I, 3): OM = 6 * (D$ = ",X")
+ 7 * (D$ = ",Y"): IF OM = 0 THEN 3000
600 IF (OM = 2 OR OM = 3) AND LEFT$ (AS$, 1)
= "B" AND AS$ < > "BIT" THEN OM = 11
610 FOR I = 1 TO 256: IF AS$ = OP$(I) THEN
IF O$(I) = OM THEN 630
620 NEXT I: GOTO 3000
630 OC = I - 1
790 REM ** TULOSTUS **
800 X = 1: FOR I = 1 TO L + 4: PRINT H$: NEXT
I: IF OM = 11 THEN 900
810 ON OM GOTO 840, 820, 840, 850, 850, 840,
840, 840, 820, 820, 840, 820, 840
820 X = X + 1: Y% = AR / 256: AR = AR - Y% *
256: MA = PC + 2: DT = Y%: GOSUB 9000: DE = Y%:
GOSUB 80: PRINT HE$: H$: H$:
840 X = X + 1: MA = PC + 1: DT = AR: GOSUB
9000: DE = AR: GOSUB 80: PRINT H$: H$: HE$:
850 MA = PC: DT = OC: GOSUB 9000: DE = OC: GOSUB
80: PRINT H$: H$: H$: H$: H$: HE$: PC = PC + X: GOTO
345
860 IF LEFT$ (HE$, 1) = "$" THEN HE$ =
RIGHT$ (HE$, LEN (HE$) - 1): GOSUB 40: RETURN
870 DE = VAL (HE$): RETURN
900 T = AR - PC - 2: IF T < - 128 OR T > 127
THEN PRINT : PRINT " LIIAN PITKA HYPPIY!": GOTO
345
910 IF T < 0 THEN T = T + 256
920 AR = T: GOTO 810
1000 Y = L - I: X = 4 - HE: FOR F = 0 TO + X *
(X < = Y) + Y * (Y < X): IF L < I THEN 1035
1010 E = ASC ( MID$ (IN$, I + F, 1))
1020 IF E > 47 AND E < 58 OR E > 64 AND E <
71 AND HE THEN NEXT
1030 IF HE = 1 AND F < > 2 AND F < > 4 THEN
F = 0
1035 IF F = 0 THEN RETURN

```

```

1040 HE$ = MID$(IN$, I - HE, F + HE): I = I +
F: GOSUB 860: AR = DE: RETURN
3000 PRINT TAB( 30); "VIRHE !": GOTO 345
3010 PRINT TAB( 30); "VIRHE MNE": GOTO 345
3020 PRINT TAB( 30); "VIRHE DSM": GOTO 345
3999 REM
4000 PRINT : PRINT : PRINT "DISASSEMBLY "
4005 GOSUB 4960: IF I = L THEN DC = AO: GOTO
4007
4006 HE$ = N$: GOSUB 860: DC = DE
4007 INPUT "RUUDULLE (R) VAI KIRJOITTIMELLE
(K) >"; F$: PRINT
4009 IF F$ = "R" THEN FA = 0: GOTO 4015
4010 IF F$ < > "K" THEN 4007
4012 PRINT D$; "PR#1": FA = 1
4015 DE = DC: GOSUB 90: PRINT "$": HE$; " ";
4020 MA = DC: GOSUB 9500: OC = DT: AS$ = OP$(OC
+ 1): OM = O%(OC + 1)
4025 DE = OC: GOSUB 80: OC$ = HE$: IF OM = 11
THEN 4340
4150 IF OM = 4 THEN X = 1: GOTO 4185
4160 MA = DC + 1: GOSUB 9500: DE = DT: GOSUB
80: X$ = HE$: MA = DC + 2: GOSUB 9500: DE = DT:
GOSUB 80: Y$ = HE$
4170 OC$ = OC$ + " " + X$: X = 2
4180 ON OM GOSUB 4210, 4220, 4230, 4230,
4230, 4260, 4270, 4280, 4290, 4300, 4310, 4320,
4330
4185 TB$ = "": FOR Z = 1 TO 20 - LEN
(OC$): TB$ = TB$ + " ": NEXT
4190 PRINT OC$: TB$: AS$: DC = DC + X
4193 GOSUB 7500: IF FLAG = 0 THEN 4015
4194 IF A$ = "X" AND FA = 1 THEN PRINT
D$; "PR# 0": GOTO 330
4195 IF A$ = "X" THEN 330
4198 GET A$
4200 GOTO 4015
4210 AS$ = AS$ + " #" + X$: RETURN
4220 OC$ = OC$ + Y$ + " ": AS$ = AS$ + " $" +
Y$ + X$: X = 3: RETURN
4230 AS$ = AS$ + " $" + X$: RETURN
4260 AS$ = AS$ + " ($" + X$ + ", X)": RETURN
4270 AS$ = AS$ + " ($" + X$ + ")", Y$: RETURN
4280 AS$ = AS$ + " $" + X$ + ", X": RETURN
4290 GOSUB 4220: AS$ = AS$ + ", X": RETURN
4300 GOSUB 4220: AS$ = AS$ + ", Y": RETURN
4320 OC$ = OC$ + Y$ + " ": AS$ = AS$ + " ($" +
Y$ + X$ + "): X = 3: RETURN
4330 AS$ = AS$ + " $" + X$: RETURN
4340 MA = DC + 1: GOSUB 9500: OF = DT: DE = OF:
GOSUB 80: OC$ = OC$ + " " + HE$
4350 DE = DC + OF + 2: IF OF > 127 THEN DE =
DE - 256
4355 GOSUB 90: AS$ = AS$ + " $" + HE$
4360 X = 2: GOTO 4185

```



```

4399 REM
4400 PRINT : PRINT "MONITORI"
4410 GOSUB 4960: IF I = L THEN 3000
4415 HE$ = N$: GOSUB 860: DC = DE
4420 PRINT : DE = DC: GOSUB 90: PRINT "$": HE$;
4425 MA = DC: GOSUB 9500: DE = DT: GOSUB 80:
PRINT " $": HE$; "/";
4430 IN$ = ""
4440 GET A$
4445 IF A$ = CHR$ (27) THEN PRINT : RETURN
: REM ESC: POIS MONITORISTA
4450 IF A$ = CHR$ (13) THEN 4510: REM CR :
SEURAAVA OSOITE
4460 IF A$ = CHR$ (9) THEN DC = DC - 1: GOTO
4420: REM TAB: TAVU TAAKSEPAIN
4470 IF A$ = H$ AND LEN (IN$) > 1 THEN IN$ =
LEFT$ (IN$, LEN (IN$) - 1): PRINT H$;: GOTO
4440
4475 IF A$ = H$ AND LEN (IN$) = 1 THEN IN$ =
"": PRINT H$;
4476 IF A$ = H$ THEN GOTO 4440
4480 A = ASC (A$): IF A < 48 AND A < > 36 OR
A > 70 OR A > 57 AND A < 65 THEN 4440
4490 IF LEN (IN$) = 3 THEN 4440
4500 IN$ = IN$ + A$: PRINT A$;: GOTO 4440
4510 IF IN$ = "" THEN 4540
4520 IF LEN (IN$) = 1 THEN 4440
4530 HE$ = IN$: GOSUB 860: IF DE < 256 THEN MA
= DC: DT = DE: GOSUB 9000
4540 DC = DC + 1: GOTO 4420
4598 REM
4599 REM ** .GO **
4600 PRINT : GOSUB 4960: IF I = L THEN MA =
AO: GOSUB 9900: GOTO 330
4605 I = I + 1: GOSUB 475: IF I$ = "$" THEN I
= I + 1: HE = 1
4610 GOSUB 1000: IF F = 0 THEN 3000
4620 MA = AR: GOSUB 9900: GOTO 330
4698 REM
4699 REM ** .ASM **
4700 GOSUB 4960: IF I = L THEN 3000
4720 I = I + 1: GOSUB 475: IF I$ = "$" THEN I
= I + 1: HE = 1
4725 GOSUB 1000: IF F = 0 THEN 3000
4730 PC = AR: PRINT : PRINT : GOTO 345
4798 REM
4799 REM ** SAVE **
4800 GOSUB 4960: IF I = L THEN PRINT TAB(
30): "TIEDOSTONNIMI PUUTTUU!!": RETURN
4802 SA = AO: SL = PC
4805 PRINT : INPUT "ALKUOSOITE: "; HE$: IF HE$
< > "" THEN GOSUB 860: SA = DE
4810 INPUT "LOPPUOSOITE: "; HE$: IF HE$ < > ""
THEN GOSUB 860: SL = DE + 1
4820 PRINT "SAVING..."

```



```

4830 PRINT D$;"BSAVE ";N$;"A";SA;"L";SL -
SA
4840 RETURN
4898 REM
4900 REM ** LOAD **
4910 GOSUB 4960: IF I = L THEN PRINT TAB(
30);"TIEDOSTONNIMI PUUTTUU!!": RETURN
4920 PRINT
4930 PRINT "LOADING..."
4940 PRINT D$;"BLOAD";N$
4950 PRINT : RETURN
4960 FOR I=1 TO L - 1: GOSUB 475: IF I$ = " "
THEN 4980
4970 NEXT I
4980 IF L = I THEN N$ = "": RETURN
4990 N$ = RIGHT$(IN$,L - I): RETURN
5000 DATA BCC, 144, BCS, 176, BEQ, 240, BMI,
48, BNE, 208, BPL, 16, BVC, 80, BVS, 112, BRK, 0
5010 DATA CLC, 24, CLD, 216, CLI, 88, CLV,
184, DEX, 202, DEY, 136, INX, 232, INY, 200,
NOP, 234
5020 DATA PHA, 72, PHP, 8, PLA, 104, PLP, 40,
RTI, 64, RTS, 96, SEC, 56, SED, 248, SEI, 120
5030 DATA TAX, 170, TAY, 168, TSX, 186, TXA,
138, TXS, 154, TYA, 152, JSR, 32, JMP, 76, JMP,
108
5040 DATA LDX, 182, STX, 150
5050 DATA ADC, 105, AND, 41, CMP, 201, EOR,
73, LDA, 169, ORA, 9, SBC, 233
5060 DATA STA, 0, 141, 133, 0, 0, 129, 145, 149, 157,
153
5070 DATA ASL, 0, 14, 6, 10, 0, 0, 0, 22, 30, 0
5080 DATA BIT, 0, 44, 36, 0, 0, 0, 0, 0, 0, 0
5090 DATA CPX, 224, 236, 228, 0, 0, 0, 0, 0, 0, 0
5100 DATA CPY, 192, 204, 196, 0, 0, 0, 0, 0, 0, 0
5110 DATA DEC, 0, 206, 198, 0, 0, 0, 0, 214, 222, 0
5120 DATA INC, 0, 238, 230, 0, 0, 0, 0, 246, 254, 0
5130 DATA LDX, 162, 174, 166, 0, 0, 0, 0, 0, 0, 190
5140 DATA LDY, 160, 172, 164, 0, 0, 0, 0, 180, 188, 0
5150 DATA LSR, 0, 78, 70, 74, 0, 0, 0, 86, 94, 0
5160 DATA ROL, 0, 46, 38, 42, 0, 0, 0, 54, 62, 0
5170 DATA ROR, 0, 110, 102, 106, 0, 0, 0, 118, 126, 0
5180 DATA STX, 0, 142, 134, 0, 0, 0, 0, 0, 0, 0
5190 DATA STY, 0, 140, 132, 0, 0, 0, 0, 148, 0, 0
6095 REM ** ALOITUSRUUTU **
6100 HOME : PRINT TAB( 8);"** MikroASSEMBLER
**"
6110 PRINT : PRINT
6120 PRINT TAB( 7);"(C) EEP 1985 MB": PRINT
6130 PRINT TAB( 7);"Apple II sovitus"
6140 PRINT TAB( 7);"(C) Aki Korhonen 1985"
6150 PRINT : PRINT : PRINT
6160 PRINT TAB( 10);" Odota hetkinen ..."
6200 RETURN

```

```
7500 REM ** Onko näppäintä painettu, jos on,  
tallenna tulos A$:iin **  
7510 FLAG = 1  
7520 IF PEEK ( - 16384 ) > 128 THEN A$ =  
CHR$ ( PEEK ( - 16384 ) - 128 ) : POKE - 16368,0:  
RETURN  
7530 FLAG = 0: RETURN  
9000 REM ** POKE MA , DT **  
9010 IF MA > 32767 THEN POKE MA - 65536,DT  
9020 IF MA < = 32767 THEN POKE MA,DT  
9030 RETURN  
9500 REM ** PEEK ( MA ) = DT **  
9510 IF MA > 32767 THEN DT = PEEK ( MA -  
65536 )  
9520 IF MA < = 32767 THEN DT = PEEK ( MA )  
9530 RETURN  
9900 REM ** CALL MA **  
9910 IF MA > 32767 THEN CALL MA - 65536  
9920 IF MA < = 32767 THEN CALL MA  
9930 RETURN
```

# MikroASSEMBLER

## Atari

Atarin Basicin konekieli ohjelman käynnistävä USR-käskey poikkeaa muiden mikrojen vastaavista, sillä se pystyy välittämään useita parametreja (arvoja) konekieli ohjelmalle. Nämä parametrit Basic-tulkki tallentaa kahden tavun muodossa pinon. Pinon päällimmäiseksi se asettaa parametrien lukumäärän ja alimmaiseksi paluusoitteen Basiciin.

Mikäli parametreja ei ole, eli konekieli ohjelmaa kutsutaan suoraan (esimerkiksi USR(1536)) tallentuu pinon kolme tavua: parametrien lukumäärä, paluusoitteen vähemmän- ja enemmänmerkitsevä tavu. Mikäli ensimmäistä tavua ei poisteta pinosta, ei RTS-käskey pysty palauttamaan ohjausta takaisin Basicille. Monipuolisuudella on harminsa.

MikroASSEMBLERISSA tämä ongelma on kierretty vektoroimalla USR-kutsu muistipaikkojen 1005-1008 kautta. Näistä kaksi ensimmäistä sisältävät PLA- ja JMP-käskyt. Kahteen jälkimmäiseen POKetaan käynnistettävän ohjelman osoite.

Mikäli haluat rakentaa Basicista helposti käynnistettäviä ohjelmia voit joko poistaa vektorinnin riveiltä 4600-4640 tai aloittaa ohjelmasi NOP-käskyllä, jonka vaihdat PLA-käskyksi sen jälkeen kun ohjelmasi toimii. Parametrien välittäminen vaatii myös muutoksia edellämaituilla riveille.

MikroASSEMBLER käyttää kanavia #1 ja #2. Varsinkin #1:n käyttäminen häiritsee sen toimintaa. Käytä omissa ohjelmissasi kanavia #3, #4 ja #5.

MikroASSEMBLER tallentaa ohjelmat siten, että ohjelma alkaa kahdella \$FF:llä, sitä seuraa kahden tavun latausosoite ja kahden tavun tallennusosoite. Näin tallennettu ohjelma voidaan ladata ja käynnistää levyiltä suoraan.

```

10 REM *** MIKRO ASSEMBLER ***
11 OPEN #1,4,0,"K:"
12 DIM HS$(6),HS$(1),TS$(16),AS$(3),OP$(257*3),IN$(2
57),KS$(1),PI$(2),IS$(1),DS$(10)
13 DIM NS$(256),OC$(10),BS$(1),YS$(6),XS$(6),AS$(1),DE
S$(12)
14 HS=CHR$(126):BS=CHR$(30)
20 TS="0123456789ABCDEF":GOSUB 6095:GOTO 130
30 REM *** DES <--> HEX KAANNOKSET ***
40 DE=0:PO=1:FOR M=LEN(HS$) TO 1 STEP -1:IF M=0 TH
EN RETURN

```

```

50 FOR G=0 TO 15:IF HE$(M,M)=T$(G+1,G+1) THEN 70
60 NEXT G
70 DE=DE+PO*G:PO=PO*16:NEXT M:RETURN
80 P=2:PPROS=16:GOTO 100
90 P=4:PPROS=4096
100 HE$="":FOR M=1 TO P:SPROS=INT(DE/PPROS)
110 HE$(LEN(HE$)+1)=T$(SPROS+1,SPROS+1):DE=DE-SPRO
S*PPROS:PPROS=INT(PPROS/16):NEXT M:RETURN
120 REM *** TAULUKOIDEN ALUSTUS ***
130 DIM OPROS(256)
140 FOR I=1 TO 256:OP$(I*3,I*3+2)="???":OPROS(I)=4
:NEXT I:FOR I=1 TO 38
150 READ ASS,P:OP$(P*3+3,P*3+5)=ASS:OPROS(P+1)=11-
7*(I>8)-2*(I>33)+10*(I>35)+(I>36):NEXT I
160 FOR I=1 TO 7:READ ASS,P:FOR J=1 TO 10
170 P1=P+4*(J=2)-4*(J=3)-8*(J=6)+8*(J=7)+12*(J=8)+
20*(J=9)+16*(J=10)+1
175 OP$(P1*3,P1*3+2)=ASS:OPROS(P1)=J:IF J=3 THEN J
=5
180 NEXT J:NEXT I:FOR I=1 TO 14:READ ASS:FOR J=1 T
O 10:READ P:IF P=0 THEN 190
185 OP$(P*3+3,P*3+5)=ASS:OPROS(P+1)=J
190 NEXT J:NEXT I:PRINT "PAINA NAPIKKAA!!"
200 GET #1,A
260 REM
280 PRINT CHR$(125);"ALKUOSOITE >";:INPUT HE$:GOSU
B 860:A0=DE:PC=A0
320 PRINT CHR$(125)
330 PRINT " ** ASSEMBLY ** ":PRINT
345 IN$="":B=0:DE=PC:HE=0:Z=0:GOSUB 90:PRINT "$";H
E$;
360 GET #1,K:K$=CHR$(K)
365 IF K$=CHR$(155) THEN 380
367 IF K$=H$ AND LEN(IN$)>1 THEN IN$=IN$(1,LEN(IN$
)-1):PRINT H$;:GOTO 360
368 IF K$=H$ AND LEN(IN$)=1 THEN IN$="":PRINT H$;:
GOTO 360
369 IF K$=H$ THEN 360
370 PRINT K$;:IN$(LEN(IN$)+1)=K$:GOTO 360
380 L=LEN(IN$):IF L<2 THEN PRINT:GOTO 345
382 PI$=IN$(1,2):IF PI$=".A" THEN GOTO 4700
384 IF PI$=".D" THEN 4000
386 IF PI$=".M" THEN GOSUB 4400:GOTO 330
388 IF PI$=".S" THEN GOSUB 4800:GOTO 345
390 IF PI$=".L" THEN GOSUB 4900:GOTO 345
392 IF PI$=".G" THEN 4600
394 REM TAHAN LISAKOODIEN TARKISTUS
410 REM ** OSOITUSMUODON TARKISTUS **
420 IF L<3 THEN 3010
430 ASS=IN$(1,3)
440 IF L=3 THEN OM=4:GOTO 600
450 I=4:GOSUB 475:IF L=4 OR IS<>" " THEN 3010
470 I=5:GOSUB 475:IF IS<>"#" THEN 480
471 I=I+1:OM=1:IF L<I THEN 3000

```

```

472 GOSUB 475:IF I$="$" THEN HE=1:I=I+1
473 GOSUB 1000:IF F=0 OR F>2 AND HE=1 OR F>3 OR L>
=I THEN 3000
474 GOTO 600
475 I$=IN$(I,I):RETURN
480 IF I$="$" THEN HE=1:I=I+1
485 IF I$="(" THEN I=I+1:GOTO 550
490 GOSUB 1000:IF F=0 THEN 3000
495 IF AR>255 THEN Z=1
500 IF L<I THEN OM=3-Z:GOTO 600
510 IF L-I<1 THEN GOTO 3000
520 IF IN$(I,I+1)="," THEN OM=8+Z:GOTO 600
530 IF IN$(I,I+1)="," THEN OM=13-3*Z:GOTO 600
540 GOTO 3000
550 IF IN$(I,I)="$" THEN HE=1:I=I+1
560 GOSUB 1000:IF F=0 OR F>5 THEN 3000
570 IF L=I+2 THEN 590
580 IF IN$(I,I)="" THEN OM=12:GOTO 600
590 D$=IN$(I,I+2):OM=6*(D$=","X")+7*(D$=","Y"):IF
OM=0 THEN 3000
600 IF (OM=2 OR OM=3) AND AS$(1,1)="B" AND AS$<>"B
IT" THEN OM=11
610 FOR I=1 TO 256:IF AS$=OP$(I*3,I*3+2) THEN IF O
PROS(I)=OM THEN 630
620 NEXT I:GOTO 3000
630 OC=I-1
790 REM ** TULOSTUS **
800 X=1:FOR I=1 TO L+4:PRINT BS$;:NEXT I:IF OM=11
THEN 900
810 ON OM GOTO 840,820,840,850,850,840,840,840,820
,820,840,820,840
820 X=X+1:YPROS=INT(AR/256):AR=AR-YPROS*256:POKE P
C+2,YPROS:DE=YPROS:GOSUB 80:PRINT HE$;BS$;BS$;
840 X=X+1:POKE PC+1,AR:DE=AR:GOSUB 80:PRINT BS$;BS
$;HE$;
850 POKE PC,OC:DE=OC:GOSUB 80:PRINT BS$;BS$;BS$;BS
$;BS$;HE$:PC=PC+X:GOTO 345
860 IF HE$(1,1)="$" THEN HE$=HE$(2):GOSUB 40:RETUR
N
870 DE=VAL(HE$):RETURN
900 T=AR-PC-2:IF T<-128 OR T>127 THEN PRINT "LIIAN
PITKA HYPY!" :GOTO 345
910 IF T<0 THEN T=T+256
920 AR=T:GOTO 810
1000 Y=L-I:X=4-HE:FOR F=0 TO X*(X<=Y)+Y*(Y<X):IF L
<I THEN 1035
1010 E=ASC(IN$(I+F,I+F))
1020 IF E>47 AND E<58 OR E>64 AND (E<71 AND HE) TH
EN NEXT F
1030 IF HE=1 AND F<>2 AND F<>4 THEN F=0
1035 IF F=0 THEN RETURN
1040 HE$=IN$(I-HE,I-HE+F+HE-1):I=I+F:GOSUB 860:AR=
DE:RETURN
3000 PRINT " VIRHE !":GOTO 345

```

```

3010 PRINT " VIRHE MNE":GOTO 345
3020 PRINT " VIRHE OSM":GOTO 345
3999 REM
4000 PRINT CHR$(125); "*** DISSASSEMBLY ***":PRINT
4005 GOSUB 4960:IF I=L THEN DC=A0:GOTO 4007
4006 HE$=N$:GOSUB 860:DC=DE
4007 PRINT "RUUDULLE (R) VAI KIRJOITTIMELLE (K)":G
ET #1,K
4009 IF CHR$(K)="R" THEN DE$="E":GOTO 4014
4010 IF CHR$(K)="K" THEN DE$="P":GOTO 4014
4012 GOTO 4007
4014 OPEN #2,8,0,DE$
4015 DE=DC:GOSUB 90:PRINT #2;"$";HE$;" ";
4020 OC=PEEK(DC):AS$=OP$(OC*3+3,OC*3+5):OM=OPROS(O
C+1)
4025 DE=OC:GOSUB 80:OC$=HE$:IF OM=11 THEN 4340
4150 IF OM=4 THEN X=1:PRINT #2;OC$;" ";A
SS:GOTO 4190
4160 DE=PEEK(DC+1):GOSUB 80:X$=HE$:DE=PEEK(DC+2):G
OSUB 80:Y$=HE$
4170 PRINT #2;OC$;" ";X$;;X=2
4180 ON OM GOSUB 4210,4220,4230,4230,4230,4260,427
0,4280,4290,4300,4300,4320,4330
4190 DC=DC+X
4194 IF PEEK(764)=28 THEN POKE 764,255:CLOSE #2:GO
TO 330
4198 IF PEEK(764)<>255 THEN POKE 764,255:A=1^1^1^1
:GOTO 4198
4200 GOTO 4015
4210 PRINT #2;" " ;AS$;" "$";X$:RETURN
4220 PRINT #2;Y$;" " ;AS$;" "$";Y$;X$;X=3:IF O
M=2 THEN PRINT #2
4221 RETURN
4230 PRINT #2;" " ;AS$;" "$";X$:RETURN
4260 PRINT #2;" " ;AS$;" ($";X$;" ,X)":RETURN
4270 PRINT #2;" " ;AS$;" ($";X$;" ),Y":RETURN
4280 PRINT #2;" " ;AS$;" "$";X$;" ,X":RETURN
4290 GOSUB 4220:PRINT #2;" ,X":RETURN
4300 GOSUB 4220:PRINT #2;" ,Y":RETURN
4320 PRINT #2;Y$;" " ;AS$;" ($";Y$;X$;" )":X=3:
RETURN
4330 PRINT #2;AS$;" "$";X$:RETURN
4340 OF=PEEK(DC+1):DE=OF:GOSUB 80:PRINT #2;OC$;" "
;HE$;" "
4350 DE=DC+OF+2:IF OF>127 THEN DE=DE-256
4355 GOSUB 90:PRINT #2;AS$;" "$";HE$
4360 X=2:GOTO 4190
4399 REM
4400 PRINT CHR$(125); "*** MONITORI ***"
4410 GOSUB 4960:IF I=L THEN 3000
4415 HE$=N$:GOSUB 860:DC=DE
4420 PRINT :DE=DC:GOSUB 90:PRINT "$";HE$;

```

```

4425 DE=PEEK(DC):GOSUB 80:PRINT " $";HE$;"/";
4430 IN$=""
4440 GET #1,A:A$=CHR$(A)
4445 IF A$=CHR$(27) THEN PRINT :PRINT :RETURN
4450 IF A$=CHR$(155) THEN 4510
4460 IF A$="-" THEN DC=DC-1:GOTO 4420
4470 IF A$=H$ THEN IN$=IN$(1,LEN(IN$)-1):PRINT H$;
:GOTO 4440
4480 A=ASC(A$):IF A<48 AND A<>36 OR A>70 OR A>57 A
ND A<65 THEN 4440
4490 IF LEN(IN$)=3 THEN 4440
4500 IN$(LEN(IN$)+1)=A$:PRINT A$;:GOTO 4440
4510 IF IN$="" THEN 4540
4520 IF LEN(IN$)=1 THEN 4440
4530 HE$=IN$:GOSUB 860:IF DE<256 THEN MA=DC:DT=DE:
POKE DC,DE
4540 DC=DC+1:GOTO 4420
4598 REM
4599 REM *** .GO ***
4600 GOSUB 4960:IF I=L THEN APU=A0:GOTO 4615
4605 I=I+1:GOSUB 475:IF I$="$" THEN I=I+1:HE=1
4610 GOSUB 1000:APU=AR:IF F=0 THEN 3000
4615 POKE 1005,104:POKE 1006,76:POKE 1007,APU-INT(
APU/256)*256:POKE 1008,APU/256
4620 Q=USR(1005):PRINT :PRINT :GOTO 330
4698 REM
4699 REM *** .ASM ***
4700 GOSUB 4960:IF I=L THEN 3000
4720 I=I+1:GOSUB 475:IF I$="$" THEN I=I+1:HE=1
4725 GOSUB 1000:IF F=0 THEN 3000
4730 PC=AR:PRINT :PRINT :GOTO 345
4798 REM
4799 REM *** SAVE ***
4800 GOSUB 4960:IF I=L THEN DE$="C":GOTO 4805
4801 DE$="D:"
4805 SA=A0:PRINT :PRINT "ALKUOSOITE:";:INPUT HE$:I
F HE$<>" THEN GOSUB 860:SA=DE
4810 SL=PC:PRINT "LOPPUOSOITE:";:INPUT HE$:IF HE$<
>" THEN GOSUB 860:SL=DE
4815 START=A0:PRINT :PRINT "KAYNNISTYSOSOITE:";:IN
PUT HE$:IF HE$<>" THEN GOSUB 860:START=DE
4825 DE$(3)=N$
4827 PRINT "SAVING...";DE$
4830 OPEN #2,8,0,DE$:PUT #2,255:PUT #2,255
4832 PUT #2,SA-INT(SA/256)*256:PUT #2,SA/256
4834 PUT #2,SL-INT(SL/256)*256:PUT #2,SL/256
4836 FOR QQ=SA TO SL:PUT #2,PEEK(QQ):NEXT QQ
4837 PUT #2,226:PUT #2,2:PUT #2,227:PUT #2,2
4838 PUT #2,START-INT(START/256)*256:PUT #2,START/
256
4839 CLOSE #2
4840 RETURN
4898 REM
4900 REM *** LOAD ***

```



```

4910 GOSUB 4960:IF I=L THEN DE$="C":GOTO 4920
4915 DE$="D:"
4920 DE$(3)=N$
4921 PRINT:PRINT "LOADING...":DE$
4922 OPEN #2,4,0,DE$:GET #2,X:GET #2,X
4924 GET #2,X:GET #2,Y:LA=X+256*Y
4926 GET #2,X:GET #2,Y:LL=X+256*Y
4927 PRINT "ALKU:";LA:PRINT "LOPPU:";LL
4928 FOR QQ=LA TO LL:GET #2,X:POKE QQ,X:NEXT QQ
4929 CLOSE #2
4930 PC=LL+1:RETURN
4960 FOR I=1 TO L-1:GOSUB 475:IF I$=" " THEN 4980
4970 NEXT I
4980 IF I=L THEN N$="":RETURN
4990 N$=IN$(I+1,L):RETURN
5000 DATA BCC,144,BCS,176,BEQ,240,BMI,48,BNE,208,B
PL,16,BVC,80,BVS,112,BRK,0
5010 DATA CLC,24,CLD,216,CLI,88,CLV,184,DEX,202,DE
Y,136,INX,232,INY,200,NOP,234
5020 DATA PHA,72,PHP,8,PLA,104,PLP,40,RTI,64,RTS,9
6,SEC,56,SED,248,SEI,120
5030 DATA TAX,170,TAY,168,TSX,186,TXA,138,TXS,154,
TYA,152,JSR,32,JMP,76,JMP,108
5040 DATA LDX,182,STX,150
5050 DATA ADC,105,AND,41,CMP,201,EOR,73,LDA,169,OR
A,9,SBC,233
5060 DATA STA,0,141,133,0,0,129,145,149,157,153
5070 DATA ASL,0,14,6,10,0,0,0,22,30,0
5080 DATA BIT,0,44,36,0,0,0,0,0,0,0
5090 DATA CPX,224,236,228,0,0,0,0,0,0,0
5100 DATA CPY,192,204,196,0,0,0,0,0,0,0
5110 DATA DEC,0,206,198,0,0,0,0,214,222,0
5120 DATA INC,0,238,230,0,0,0,0,246,254,0
5130 DATA LDX,162,174,166,0,0,0,0,180,190
5140 DATA LDY,160,172,164,0,0,0,0,180,188,0
5150 DATA LSR,0,78,70,74,0,0,0,86,94,0
5160 DATA ROL,0,46,38,42,0,0,0,54,62,0
5170 DATA ROR,0,110,102,106,0,0,0,118,126,0
5180 DATA STX,0,142,134,0,0,0,0,0,0,0
5190 DATA STY,0,140,132,0,0,0,0,148,0,0
6095 REM *** ALOITUSRUUTU ***
6100 PRINT CHR$(125):"*** MIKROASSEMBLER ***"
6120 PRINT "(C) EEP 1985 MB":PRINT
6130 PRINT "Atari sovitus Petri Helenius & kl":PRI
NT
6140 PRINT " MIKROASSEMBLER SIJOITTAA ANTAMAASI
MNEMONISTA KASKYA VASTAAVAT KOODIT SUORAAN MUIS
TIIN."
6150 PRINT " MUUT TOIMINNOT VOIT KAYNNISTAA
ANTAMALLA KOODIN SIJAAN TAULUKON MUKAISEN PIS
TEKOODIN."
6160 PRINT "PISTE JA ENSIMMAINEN KIRJAIN RIITTA.
PALUU ASSEMBLERIIN ESC-NAPPAIMELLA."
6170 PRINT ".ASM ALOITA UUDESTA OSOITTEESTA"

```



```
6180 PRINT " .DIS  DISASSEMBLER/KIRJOITINLISTAUS"
6190 PRINT " .MON  TUTKI JA MUUTA MUISTIPAIKKOJA"
6200 PRINT " .SAV  TALLENNA NAUHALLE/LEVYLLE"
6210 PRINT " .LOA  LATAA KOODI NAUHALTA/LEVYLTA"
6220 PRINT " .GO   KAYNNISTA KONEK. OHJELMA"
6240 PRINT :PRINT "ODOTA HETKI..." ;CHR$(28)
6250 RETURN
```

# Ohjelmien syöttöohjeet

Tietokoneet ovat pikkutarkkoja laitteita. Ohjelmia kirjoitettaessa on pidettävä erityistä huolta siitä, että ohjelmat kirjoitetaan täsmälleen kuten listauksessa mukaanlukien pisteet, pilkut ja muut välimerkit.

Listauksissa ohjelmarivi on tulostettu useammalle peräkkäiselle riville. Ohjelmaa syötettäessä yksi käskyrivi syötetään kuitenkin kokonaisuena ja RETURN-näppäintä (rivisiirtoa) painetaan vasta koko ohjelmarivin syötön jälkeen.

Ennenkuin alat kirjoittaa ohjelmia koneeseesi, tutustu huolella sen ohjekirjaan ja perehdy tarkasti mikrosi ominaisuuksiin. Opettele käyttämään näppäimistöä ja varsinkin editointi- eli korjailunäppäimiä. Niiden käytön opettelemiseen uhratun ajan voittaa nopeasti takaisin. Erikoisesti kursorinäppäimien (kohdistimen siirto), deleten (poista merkki) ja insertin (lisää merkki) toiminnan ymmärtäminen on tärkeää.

Opettele, kuinka ohjelmat tallennetaan kasetille tai levykkeelle ja kuinka ne ladataan takaisin koneeseen. Tallenna kirjoittamasi ohjelma aina ennen kuin yrität koeajaa sitä. Jos ohjelma on pitkä, voit kirjoittaa sen useammassakin erässä. Keskenäisenkin ohjelman voi tallentaa kasetille tai levykkeelle. Jatkaessasi ohjelman kirjoittamista, lataat vain jo kirjoitetun osuuden muistiin ja jatkat syöttöä seuraavasta rivinumerosta.

Koska useat CBM-64:n ja VIC-20:n ohjelmat sisältävät vaikealukuisia ja vaikeasti kirjoitettavia erikoismerkkejä, olemme listanneet niiden ohjelmat luettavammassa muodossa. Jokaisen erikoismerkin tilalle on listaukseen laitettu merkin nimitys lyhennetyssä muodossa. Nimitys on useimmiten sama kuin se, joka löytyy näppäimestä.

Lyhennykset on sijoitettu kulmasulkujen väliin. Esimerkiksi RVSON-näppäin, eli CONTROL+9, joka tavalliseen tulostukseen ja kuvaruudulle kirjoittaisi käänteisen R-kirjaimen, on listauksissa selvennetty merkinnällä <RVSON>. Samaten viisi kursorin siirtoa vasemmalle on selvennetty merkinnällä <5LEFT>.

| Listauksessa | Näppäile         | Kuvaruudulla                                                                        |
|--------------|------------------|-------------------------------------------------------------------------------------|
| <CLR>        | SHIFT + CLR/HOME |    |
| <HOME>       | CLR/HOME         |    |
| <UP>         | SHIFT + ↑ CRSR ↓ |    |
| <DOWN>       | ↑ CRSR ↓         |    |
| <LEFT>       | SHIFT + ← CRSR → |    |
| <RIGHT>      | ← CRSR →         |    |
| <RVSON>      | CTRL + 9         |    |
| <RVSOFF>     | CTRL + 0         |    |
| <BLK>        | CTRL + 1         |    |
| <WHT>        | CTRL + 2         |    |
| <RED>        | CTRL + 3         |    |
| <CYN>        | CTRL + 4         |    |
| <PUR>        | CTRL + 5         |    |
| <GRN>        | CTRL + 6         |    |
| <BLU>        | CTRL + 7         |   |
| <YEL>        | CTRL + 8         |  |
| <F1>         | f1               |  |
| <F2>         | f2               |  |
| <F3>         | f3               |  |
| <F4>         | f4               |  |
| <F5>         | f5               |  |
| <F6>         | f6               |  |
| <F7>         | f7               |  |
| <F8>         | f8               |  |

Commodore 64:n ja VIC-20:n listauksissa käytetyt selvennykset.

# Tarkastaja (Commodore 64 ja VIC-20)

Jotta listausten sisäänsyöttö tapahtuisi varmasti ilman virheitä, olemme kehittäneet Commodore 64:lle ja VIC-20:lle Tarkastaja-ohjelman, joka rivi riviltä valvoo syöttöä ja ilmoittaa mahdollisista virheistä.

Tarkastaja on konekielinen ohjelma, joka on jatkuvasti toiminnassa syöttäessäsi Basic-listausta koneeseen. Kun painat RETURN-näppäintä (rivinsiirto), tulostaa Tarkastaja kuvaruudun ylälaitaan kaksinumeroisen tarkistussumman. 64:n ja VIC-20:n listauksissa on jokaisen ohjelmarivin lopussa REM-lause, jossa on myös kaksinumeroinen heksaluku. Jos Tarkastajan ilmoittama tarkistussumma on erilainen kuin listauksessa oleva luku, on syötössä tapahtunut virhe.

Kirjoita nyt Tarkastaja-ohjelma koneeseesi ja tallenna se kasetille tai levykkeelle tulevaa käyttöä varten. Tallennuksen jälkeen voit koeajaa Tarkastajan komennolla RUN. Jos kuvaruudulle tulostuu teksti 'VIRHE DATOISSA', olet tehnyt syöttövirheen. Se on korjattava ja ohjelma tallennettava uudelleen.

Ennen kuin alat syöttää listausta koneeseesi esimerkiksi tästä kirjasta, lataa Tarkastaja koneesi muistiin ja aja se komennolla RUN. Kun kuvaruudulle ilmestyy teksti 'TARKASTAJA OK.', on oikolukuohjelma käyttövalmis. Kirjoita vielä NEW, joka poistaa Tarkastajan Basic-kielisen osuuden ja ala kirjoittaa halua- maasi listausta.

Älä kirjoita rivin lopussa olevaa REM-lausetta lainkaan! Tarkistussummasta tulee muuten virheellinen. Jos tarkistussummaa ei tule kuvaruudulle näkyviin RETURNin painamisen jälkeen, siirrä kohdistin uudestaan juuri syötetyn rivin päälle ja paina RETURNia. Nyt tarkistussumman pitäisi ilmestyä kuvaruudun oikeaan yläkulmaan.

```
0 FOR A=820 TO 827: READ B: POKEA,B: C=C+B: NEXT
1 FOR A=679 TO 750: READ B: POKEA,B: C=C+B: NEXT
2 IF C<>8484 THEN PRINT "VIRHE DATOISSA.":END
3 SYS 679: PRINT "<CLR>TARKASTAJA OK.": END
5 DATA 201,186,144,3,56,233,57,96
6 DATA 169,2,141,37,3,169,178,141,36
7 DATA 3,96,32,87,241,133,252,8,201
8 DATA 32,240,47,201,13,208,38,165,253
9 DATA 41,15,9,176,32,52,3,141,42
10 DATA 4,165,253,74,74,74,74,9,176
11 DATA 32,52,3,141,41,4,169,0,141
12 DATA 41,216,141,42,216,169,0,240,3
13 DATA 24,101,253,133,253,165,252,40,96
READY.
```

*Tarkastajaohjelma Commodore 64:lle.*

```
1 FOR A=685 TO 703: READ B: POKEA,B: C=C+B: NEXT
2 FOR A=722 TO 764: READ B: POKEA,B: C=C+B: NEXT
3 FOR A=820 TO 827: READ B: POKEA,B: C=C+B: NEXT
4 IF C<>7411 THEN PRINT "VIRHE DATOISSA.": END
5 SYS685:PRINT "TARKASTAJA OK.": END
685 DATA 169,2,141,37,3,169,210,141,36,3,96
696 DATA 24,101,253,133,253,104,40,96
722 DATA 32,14,242,8,72,201,32,240,226,201,13
733 DATA 208,217,165,253,41,15,9,176,32,52,3,141
745 DATA 23,30,165,253,74,74,74,74,9,176
755 DATA 32,52,3,141,22,30,169,0,240,190
820 DATA 201,186,144,3,56,233,57,96
READY.
```

*Tarkastajaohjelma laajentamattomalle VICille ja  
superexpanderilla varustetulle VICille.*

```
1 FOR A=685 TO 703: READ B: POKEA,B: C=C+B: NEXT
2 FOR A=722 TO 764: READ B: POKEA,B: C=C+B: NEXT
3 FOR A=820 TO 827: READ B: POKEA,B: C=C+B: NEXT
4 IF C<>7383 THEN PRINT "VIRHE DATOISSA.": END
5 SYS685:PRINT "TARKASTAJA OK.": END
685 DATA 169,2,141,37,3,169,210,141,36,3,96
696 DATA 24,101,253,133,253,104,40,96
722 DATA 32,14,242,8,72,201,32,240,226,201,13
733 DATA 208,217,165,253,41,15,9,176,32,52,3,141
745 DATA 23,16,165,253,74,74,74,74,9,176
755 DATA 32,52,3,141,22,16,169,0,240,190
820 DATA 201,186,144,3,56,233,57,96
READY.
```

*Tarkastajaohjelma vähintään 8 kilotavun lisämuistilla  
varustetulle VICille.*

# HAKEMISTO

## A

Absolute addressing,  
    kts. absoluuttinen osoitus  
Absolute,X addressing,  
    kts. absoluuttinen,X -osoitus  
Absolute,Y addressing,  
    kts. absoluuttinen,Y -osoitus  
absoluuttinen osoite 200  
absoluuttinen osoitusmuoto 51, 199  
absoluuttinen,X -osoitus 52, 200  
absoluuttinen,Y -osoitus 52, 200  
Accumulator addressing, kts. akkuosoitus  
accumulator, kts. akku  
ADC 44, 71-75, 166, 192, 193  
akku 19, 35, 117, 130  
akkuosoitus 62, 199  
algoritmi 96  
aliohjelma 41, 96, 100, 102  
alkurutiini 95, 96  
aloitusosoite 30, 100  
ALU 35, 46, 73  
AND 76, 166, 190, 191  
Antic, Atari 153-154  
argumentti 25, 40, 102, 165  
aritmeettis-looginen yksikkö 35, 46, 73  
ASCII -koodi 97, 101, 204  
ASL 81, 167, 190  
assembler 26, 29, 94, 103, 198  
assembler-kääntäjä 29, 102  
assemblerin käyttö 102  
assembly-kieli 26, 102  
assemblylistaus 100  
ATASCII 149, 153

## B

basic-alue, C-64 107  
Basic-ohjelmatilan alkuosoite  
    C-64 117  
    VIC-20 128  
    Apple 142  
    Atari 162  
Basic-ohjelmatilan loppuosoite  
    C-64 117  
    VIC-20 129  
    Apple 135, 142  
    Atari 147  
Basic-tulkki 12, 33, 105, 118  
Basiclataaja 28  
Basicloader, kts. Basiclataaja  
BCC 45, 85, 167, 193  
BCD-luvut 75, 90-92, 98  
BCS 45, 85, 168, 194  
BEEB, Apple 143  
BEQ 44, 85, 168, 196  
binaariluku 16, 91  
Binary coded decimal, kts. BCD-luvut  
BIT 168, 190, 191  
bitti 17, 76-79  
bittikoodi 24  
blokki 115  
BMI 85, 169, 191  
BNE 44, 85, 169, 195  
BPL 86, 169, 190  
BRK 86, 89, 170, 189, 204  
BVC 86, 170, 191  
BVS 86, 170, 192

**C**

C-lippu 44, 70, 72-75, 90  
CALL 21  
CHRGET, VIC-20 129  
CHRGET, C-64 117  
CHRIN, C-64 100, 118  
CHRIN, VIC-20 127, 131  
CHROUT, Apple 144  
CHROUT, C-64 112, 118  
CHROUT, VIC-20 131  
CIO (Central I/O), Atari 152, 158  
CLC 90, 171, 190  
CLD 171, 195  
CLI 171, 192  
CLV 172, 194  
CMP 45, 79, 172, 194, 195  
CPX 80, 173, 195, 196 CPY 80, 173, 194, 195

**D**

D-lippu 75, 90, 96, 97, 165  
data 24, 202  
DATA-lause 103  
dataväylä 39  
debugaus, kts. virheen etsintä  
debugger 103  
DEC 82, 174, 195  
dekrementointi 37  
DEX 82, 174, 195  
DEY 82, 174, 193  
DIM 24  
disassembler 37, 103, 197, 202  
display list, Atari 153-156

**E, F**

ehdoton TAI 78  
EOR 78, 175, 191, 192  
epäsuora osoitus 58, 107, 112, 200  
epäsuora,X -osoitus 56, 199  
epäsuora,Y -osoitus 55, 199  
Exclusive OR, kts. EOR  
FOR...NEXT 38, 82

**G**

GETIN, C-64 97, 119  
GETIN, VIC-20 127, 132  
GETLN, Apple 142, 144  
GOSUB 93  
GOTO 58  
GR, Apple 143  
grafiikka, Atari 153  
grafiikkatilat, Atari 148, 155  
GTIA-piiri, Atari 150

**H**

haarautuminen 84  
haarautuminen eteenpäin 62  
haarautumisosoite 60, 200  
heksadesimaaliluvut 15  
heksadumppi 27  
heksakoodi 25  
heksalataaja 27  
heksaluku 189  
heksanumero 102  
hexloader, kts. heksalataaja  
HIMEM, Apple 135, 142  
HIRES, Apple 140, 142  
HOME, Apple 137, 144  
hyppykäsky 84  
hyppyosoite 29  
hyppytaulukko 118

**I**

I-lippu 88, 90  
IF...THEN 70, 85  
Immediate addressing, kts. välitön osoitus  
Implied addressing, kts. sisältyvä osoitus  
INC 82, 175, 195, 196  
indeksi 39, 54  
indeksirekisteri 39, 67, 97  
Indirect addressing, kts. epäsuora osoitus  
Indirect,X addressing, kts. epäsuora,X -osoitus  
Indirect,Y addressing, kts. epäsuora,Y -osoitus  
inkrementointi 37  
INPUT, Apple 144  
INX 82, 176, 195  
INY 82, 176, 195  
IOCB, Atari 157  
IRQ 46, 87  
itsetesti, Atari 147

**J**

JA 76  
jakolasku 47  
JMP 40, 58, 84, 176, 191, 192, 202  
JSR 26, 42, 84, 93, 100, 118, 177, 190

**K**

kaksijärjestelmä 16  
kantaluku 16  
kasettipuskuri  
    C-64 107, 118  
    VIC-20 123, 129, 130  
    Atari 147, 163  
kellojakso 165

kellolaskuri  
     C-64 119  
     VIC-20 129  
     Atari 161  
 KERNAL 108, 116, 118, 131  
 kertolasku 47  
 keskeytykset 46, 86, 105, 108  
 keskeytyskielto 88  
 keskeytyskohta 95  
 keskeytysvektori, Atari 162  
 kilotavu 18  
 kirjallisuus 104  
     C-64 116  
     VIC-20 128  
     Apple 141  
     Atari 160  
 kohdistin 27  
 komentolohko, Atari 157  
 kommentti 30, 97, 100  
 konekielen käskyt 165-188  
 konekielen käynnistys 203  
 konekielen lataus 203  
 konekielen tallennus 203  
 konekielikoodi 102  
 konekielimonitori 103  
 konekielimonitori, Apple 144  
 konekielisen ohjelman sijoitus 24  
 "koukku" 99  
 kursori 27  
     C-64 111-112, 117  
     VIC-20 126, 129  
     Apple 143  
     Atari 161  
 kutsuosoite 104  
 kuvan vieritys, Atari 156  
 kuvaruutukoodit 98  
     C-64 113-114  
     VIC-20 126  
     Apple 138-139  
     Atari 152-153  
 kuvaruutumuisti 95, 104  
     C-64 111, 117  
     VIC-20 123, 125  
     Apple 136  
     Atari 151  
 kymmenjärjestelmä 15  
 käänteiset merkit, C-64 114, 117  
 käsin koodaus 94  
 käskykanta 65-92, 165-188  
 käskykoodi 198  
 käskylyhenne 103  
 käskyrekisteri 35, 39  
 käskyrivi 198  
 käskyryhvät 65  
 käyttöjärjestelmä 12, 100, 104

käyttöjärjestelmärutiinit, kts. ROM-rutiinit

## L

label, "leima", kts. nimiö  
 lainaus 76  
 laskentakäskyt 71  
 LDA 19, 26, 35, 55, 57, 58, 67, 111,  
     126, 152, 177, 194  
 LDX 67, 178, 194  
 LDY 67, 178, 194  
 levykäyttöjärjestelmä  
     Apple 133  
     Atari 163  
 liput 43, 90-92  
 LIST 202  
 listaus 102  
 LOAD, C-64 118  
 LOAD, VIC-20 131  
 LOMEM, Apple 142  
 loogiset operaatiot 46, 71, 76  
 lopullinen osoite 107  
 LORES, Apple 140, 142  
 LSR 63, 81, 179, 191, 192  
 lukujärjestelmät 16, 189-196  
 lähdeohjelma 102

## M

malli 97  
 maskaus 77, 78, 101, 108  
 megatavu 18  
 merkkijonon tulostus  
     C-64 112  
     VIC-20 126  
     Apple 136  
     Atari 152  
 MikroASSEMBLER 30, 197  
 mikropiiri 31  
 mikroprosessori 18  
 mnemoninen muoto, kts. käskylyhenne  
 monitori 103, 197, 202  
 muistiavaruus 41, 94  
 muistikartta 104  
     C-64 105-106  
     VIC-20 122  
     Apple 133-134  
     Atari 145-146  
 muistinlaajennus, VIC-20 121  
 muistinsiirtorutiini, Apple 143  
 muistinumero 44, 70, 72-75, 90  
 muistiosoite 20  
 muistiosoitteet  
     C-64 117  
     VIC-20 129  
     Apple 140, 142  
     Atari 161



muistipaikka 18, 32  
 muistisivu 51  
 muunnokset 189-196  
 muuttuja 11, 24, 37, 49, 94

## N

N-lippu 70  
 nimiö 62, 102, 197  
 NMI 46, 87  
 nollalippu, kts. Z-lippu  
 nollasivu 126, 147  
 nollasivu,X -osoitus 54, 199  
 nollasivu,Y -osoitus 54, 200  
 nollasivun osoitusmuoto 51, 199  
 NOP 62, 92, 179, 196  
 näppäimistö  
     C-64 108  
     VIC-20 127  
     Apple 135  
     Atari 148  
 näppäimistön esto, VIC-20 124  
 näppäimistöpuskuri  
     C-64 108  
     VIC-20 127, 129  
     Atari 149  
 näppäinkoodit 100  
     C-64 109, 117  
     VIC-20 127  
     Apple 135  
     Atari 149

## O

object, kts. konekielikoodi  
 offset, kts. haaraumisoite  
 oheislaitte 33  
 ohjainkytkin, Apple 142  
 ohjelmalaskuri 35, 39, 202  
 ohjelman kääntäminen 94  
 ohjelman korjaus 103  
 ohjelasilmukka 67  
 oktaaliluku 17  
 ORA 77, 180, 189, 190  
 osoite 25, 33  
 osoitevaruus 18  
 osoiteosa, kts. argumentti  
 osoiteväylä 39  
 osoitusmuodot 36, 49-64, 198

## P

paddlet, Apple 136  
 paluuosoite 41  
 paperihahmottelu 95  
 PC, kts. ohjelmalaskuri  
 PDL(x), Apple 136, 143

peliohjain 104  
     C-64 110  
     VIC-20 124  
     Apple 136  
     Atari 150  
 PHA 43, 69, 180, 191  
 PHP 69, 181, 190  
 pino-osoitin 35, 41, 68  
 pinomuisti 41, 68, 117, 129, 162  
 pistekoodit 201  
     .ASM 200, 201  
     .BYT 204  
     .DIS 198, 202  
     .GO 203  
     .LOA 203  
     .MON 202  
     .SAV 203  
 pistelaskuri 95-99  
 PLA 43, 69, 181, 192  
 PLOT, Apple 143  
 PLOT, C-64 112, 119  
 PLOT, VIC-20 132  
 PLP 69, 181, 190  
 POKE 19, 125, 151  
 PR , Apple 143  
 PRINT 54, 111, 125, 136, 151  
 printteripuskuri, Atari 147, 163  
 Program Counter, kts. ohjelmalaskuri  
 prosessori 18, 33  
 pysäytyskäsky 103  
 pääohjelma 96

## R

R/W-linja 46  
 RAM 31, 33  
 RDTIM, C-64 119  
 RDTIM, VIC-20 131  
 rekisteri 18, 24, 34, 95, 100, 116  
 Relative addressing, kts. suhteellinen osoitus  
 REM-lause 99  
 RETURN 84  
 RETURN-näppäin 99  
 ROL 82, 182, 190, 191  
 ROM 31, 33  
 ROM-rutiinit 04  
     C-64 119  
     VIC-20 131  
     Apple 143  
     Atari 158-160  
 ROMin poiskytkeminen  
     C-64 107  
     Atari 145, 148  
 ROR 82, 182, 192, 193  
 RTI 86, 88, 183, 191  
 RTS 30, 42, 62, 95, 183, 192, 202

RUN 33  
ruudun tyhjennys, Apple 143

## S

SAVE, C-64 119  
SAVE, VIC-20 131  
SBC 75-76, 183, 195, 196  
SCRFEN, C-64 119  
SCREEN, VIC-20 132  
SEC 90, 184, 191  
SED 184, 196  
SEI 184, 193  
SETLFS, C-64 119  
SETLFS, VIC-20 131  
SETNAM, C-64 119  
SETNAM, VIC-20 131  
SETTIM, C-64 119  
SETTIM, VIC-20 131  
siirtokäsky 66  
silmukka 96  
sisältyvä osoitusmuoto 62, 199  
sivu 51  
source, kts. lähdeohjelma  
SP, kts. pino-osoitin  
SR, kts. tilarekisteri  
ST, C-64 117  
ST, VIC-20 129  
STA 36, 67, 111, 126, 152, 185, 193  
Stack Pointer, kts. pino-osoitin  
Status Register, kts. tilarekisteri  
STX 185, 193  
STY 186, 193  
suhteellinen osoitus 59, 200  
Superexpander 121, 123  
SYS 21, 38, 100, 102  
syöttö ja tulostus, Atari 157

## T

TAB 136  
TAI 77  
tallennus 119  
Tarkastaja 99-102, 198  
tarkistussumma 28, 99  
tarkkuusgrafiikka, Apple 140  
tarkkuusgrafiikka  
C-64 115  
Apple 140  
Atari 153  
tausta-ajo 46, 99  
tavu 12, 18  
TAX 69, 186, 194  
TAY 69, 186, 194  
tekstieditori 94  
tekstin siirto 52  
testaukset 79  
TEXT, Apple 143

tietokoneen rakenne 31  
tilan varaus  
C-64 107  
VIC-20 124  
Apple 135  
Atari 148  
tilarekisteri 35, 43, 69, 118, 130  
toimintakuvaus 94, 96  
TSX 69, 187, 194  
tulitusnäppäin  
C-64 110  
VIC-20 124  
Atari 150  
tulostus kuvaruudulle  
C-64 111  
Vic-20 125  
Apple 136  
Atari 151  
TXA 69, 187, 193  
TXS 69, 187, 193  
TYA 69, 188, 193

## U, V

USR 21, 162  
vakio 24, 94  
vapaasti sijoitettavuus 61, 147  
vapaat alueet  
C-64 107  
VIC-20 123  
Apple 135  
Atari 147  
VERIFY, VIC-20 131  
VIC-II videopiiri 115  
VICin perusversio 121  
viivanpiirto, Apple 143  
virheen etsintä 62, 89, 92, 95, 103  
virheilmoitus 204  
VTAB 136  
vähennyslasku 75-76  
välitön osoitusmuoto 50, 199  
välitulosten tallennus 69  
värikoodi 97  
värimuisti 97, 101, 121, 126  
väylä 31

## X, Y, Z

X-rekisteri 35, 37, 118, 130  
Y-rekisteri 35, 37, 118, 130  
yhteenlasku 71-75  
Z-lippu 44, 69, 165  
Zero Page addressing,  
kts. nollasivun osoitus  
Zero Page,X addressing,  
kts nollasivu,X -osoitus  
Zero Page,Y addressing,  
kts. nollasivu,Y -osoitus

6502- ja 6510-prosessoreita käytetään monissa suosituissa kotitietokoneissa ohjaamaan laitteen sisäisiä toimintoja. Ne ovat periaatteeltaan muistikeskeisiä ja siksi niiden ohjelmointi konekielellä on helppo oppia. Konekielikäskyjä on 56 kappaletta ja niillä on 13 erilaista osoitusmuotoa.



"Konekielinen ohjelmointi tunkeutuu suoraan laitteen syvyysiin, alueille, jonne Basicilla ei ole asiaa. Oppiessasi konekielen pääset ohjaamaan koneen toimintaa tehokkaimmalla mahdollisella tavalla. Konekieli on koneen oma kieli."

— kirjan sisällöstä —

**BASICISTA KONEKIELEEN** on konekieliohjelmoinnin perusteos, joka ei edellytä lukijaltaan minkäänlaisia esitietoja konekielen käytöstä. Perustiedot Basicista ja ohjelmoinnin ymmärtäminen ovat kuitenkin tarpeen.

- Asiat selvitetään useiden ohjelmaesimerkkien avulla. Lukija voi välittömästi kokeilla eri käskyjä ja menetelmiä käytännössä.
- Mukana ovat myös muistikartat ja tärkeimpien ROM-rutiinien osoitteet Commodore 64:lle, Vic-20:lle, Atarille ja Applelle.
- Kirjassa on listauksena konekieliohjelmoinnin apuväline, MikroASSEMBLER, jolla konekieliohjelmaa voi rakentaa, tallentaa, ladata ja ajaa.